MICRO
FINANCE
SYSTEMS

TINY TROLL

MFS T.M.

Serial # __________

TINY TROLL REGISTRATION AND NON-DISCLOSURE AGREEMENT

Please read, complete and return one copy of this form to Micro Finance Systems (44 Parker Rd., Wellesley, MA 02181) prior to using Tiny Troll. We will advise you of enhancements to this software package and make updates available at a nominal charge for a period one year.

Both Tiny Troll and the Tiny Troll user manual are copyright 1979 by Micro Finance Systems. You may make back-up copies of this software package for your personal use. You may not make copies for use by others or for commerical purposes.

The party below agrees that all copies of Tiny Troll will be strictly safeguarded against disclosure to or use by persons not authorized by Micro Finance Systems. The party agrees that copy cr unauthorized disclosure will cause great damage to Micro Finance Systems and that this damage is far greater than the value of the software and the documentation involved.

I have read the above agreement and terms and agree to them.

__________________________ __________________________

signature date name (print or type)

____

address

MFS
MICRO
FINANCE
SYSTEMS

44 Parker Road
Wellesley, MA 02181
(617) 237-0030

October 20, 1979

TINY TROLL UPDATE

This memo should be included in the TINY TROLL manual for permanent reference. All changes refer to TINY TROLL V1.1 Check the title line of the TINY TROLL command menu to verify the release number of your diskette.

CONTENTS

1. PLOT COMMAND BUG
2. BACKING UP TINY TROLL
3. SYNTAX RESTRICTIONS
4. RUNNING DEMONSTRATION PROGRAMS
5. PRINTER INTERFACES

#1: PLOT COMMAND BUG DOES NOT APPLY TO TINY TROLL V1.2

~~The PLOT routine may "hang up" during the printing of X- or Y-axis labels under certain circumstances. This is due to a discovered error in Applesoft Basic's ONERR GOTO statement. The following fix to TINY TROLL, which you may make yourself, will correct this problem.~~

~~Procedure: Boot TINY TROLL. Type END as the first command to exit to Applesoft. Make the following changes to the Basic program in memory at this point.~~

```
Line 6:    IF EL=1 THEN GOSUB 3120: GOTO 170
Line 1900: CALL 768: Y=PEEK(222): IF Y<15 THEN
           PRINT CHR$(7);: GOSUB 2800: PRINT:
           PR=3: C$(C)=" ":GOTO 2020
Line 2010: delete this line
Line 2020: A=FRE(0): CALL 816: GET A$: RUN
```

~~Notes: Line 6 is a new line which is exactly the same as the old line 2010, which is to be deleted. Line 1900 is exactly the same as it was, except that the last statement which was GOTO 170 becomes GOTO 2020. Line 2020 should be entered exactly as shown.~~

~~N.B. Before attempting this fix (or any other modification to TINY TROLL) be sure to back up the Basic program, called TINY TROLL, which comes on the diskette. See the next section of this memo for more information on this topic.~~

Once you have made the changes to the program lines, you may save the program under the name TINY TROLL, as follows:

```
]SAVE TINY TROLL
```

Final note: In line 2020, as shown, that is a zero, not the letter 'O' after FRE(.

When TINY TROLL now finds an error, it will print an appropriate error message and <u>pause</u> until you press the space bar (or any other key).

#2: <u>BACKING UP TINY TROLL</u>

A security procedure has been applied to all TINY TROLL diskettes to prevent unauthorized duplication. This is a regrettable necessity, due to the widespread practice of software bootlegging.

We encourage you, however, to back up the Basic portion of TINY TROLL. To do this, simply boot your system using the TINY TROLL diskette, type END as the first command to exit to Basic, insert another initialized diskette into the drive and issue the command

```
]SAVE TINY TROLL
```

To restore a copy of TINY TROLL to the original diskette, use the following procedure. Boot the system from any DOS 3.2 diskette. From the diskette containing the back-up copy, issue the command

```
]LOAD TINY TROLL
```

Now, put the TINY TROLL system diskette (the one you purchased) into the drive, and type the command

```
]SAVE TINY TROLL
```

In other words, making back-up copies of the Basic program TINY TROLL is just like backing up any other program. Using a back-up procedure will insure that you can always recover a working version of TINY TROLL. This is particularly important if you modify the Basic portion of TINY TROLL in any way. Make a back-up copy (or two) before attempting any changes.

Defective TINY TROLL disks may be exchanged with the dealer from whom TINY TROLL was purchased.

#3: SYNTAX RESTRICTIONS

The PR# command to turn on the printer cannot be successfuly issued from within DEDIT mode. To print out an editing session, issue the proper PR# command before the DEDIT command.

When using the DO command to transform data, use the command form

```
DO filename = expression
```

all entered on one line. Do not press RETURN after 'DO' or the filename.

Similarly, when using any of the edit sub-commands (ADD, ERASE, or REPLACE), of the form

```
Verb date value ...
ADD  78:1 100 200 300
```

do not attempt to separate the command by typing a return after 'ADD' or the date. The example shown above is correct.

#4: RUNNING DEMONSTRATION PROGRAMS

To run the demonstration programs which are supplied on each TINY TROLL diskette, use the following procedure: Boot TINY TROLL. Type END as the first command. From Basic, enter:

```
RUN TT DEMO HEADER
```

The demonstration programs menu will automatically be invoked. Select the particular demonstration program you wish to view from the menu.

#5: PRINTER INTERFACES

Apple's communication card, when used as a printer interface, does not support the HTAB command used to format printed output. Use of the general purpose serial interface card is recommended. Parallel interfaces will also work.

TINY TROLL ADDITIONAL DOCUMENTATION

Since the original TINY TROLL manual was written, we have made a number of improvements to the system. This additional documentation describes these and also gives corrections to the manual. These are:

(1) Simplified Syntax
(2) RESET recovery
(3) Non-supported features
(4) Backing Up TINY TROLL

Pages in the manual which are affected by these changes have been so marked.

SIMPLIFIED SYNTAX

1. In general, it is not necessary to end a command with a colon (:). Simply type your TINY TROLL command. When finished, press RETURN.

2. Keywords may be abbreviated to 2 letters. In DEDIT (data edit) mode, you may use 1 letter abbreviations. (A - ADD, R - REPLACE, E - ERASE, L - LIST, Q - QUIT)

3. For your convenience you may always refer to the last file name you referenced by typing * (e.g. LOAD FILE1: PLOT * will plot the file named FILE1).

If you enter more than one complete command on a line, each command must be separated by a colon.

3. Spaces (blanks) in command lines may be freely inserted or not, as you wish. Ignore all references in the manual about the use of spaces in a DO command. The command DO A = B + C is completely equivalent to DO A=B+C. You will want to leave spaces between command keywords and filenames. This is the natural way one would type and is mentioned here on the completeness' sake. The command PLOT FILE: is correct. The command PLOTFILE: will not work (and looks silly too).

The above changes affect the following sections of the manual:

p. 14 - underlined sections

p. 21 - TABLE 1 Basic Syntactic Rules for TINY TROLL

pp. 24-25 COMMAND FORMATS, especially points 2 and 3 of TABLE 3 COMMAND FORMAT SUMMARY

RESET RECOVERY

If, at any time during a TINY TROLL session you should accidentally press the RESET key, you may resume the session without losing any of the active data files. Similarly, if the program should "hang" for any reason, you can get back to TINY TROLL without starting from the beginning again. (We have carefully removed bugs and errors from the program, but inevitably some still remain. One symptom of a bug is that the system will just sit there for a long time and do nothing. In this case, you may intentionally press the RESET key and recover the program using the procedure given below.)

Here is what to do after your press RESET. Just type RUN. The main command menu and the prompt "COMMAND>" should appear on the screen. Verify that your files are still there with the NAMES command. If they are, you have successfully recovered. (If you have an Apple II without an Auto-Start ROM, a RESET will place you in the system monitor (a * prompt will appear). In this case, return to Applesoft Basic by typing 3D0G, then type RUN.)

This recovery procedure will work 95% of the time. Occasionally, it will fail, and typing RUN will produce the command menu but not the command prompt. In this case it is necessary to re-start TINY TROLL by pressing RESET (returning to Basic if necessary) and then typing RUN 50000. RUN 50000 provides a cold-start entry to TINY TROLL. Any active data files will be lost and must be LOADed from the disk.

In extreme cases, you can always re-boot your system with the TINY TROLL disk, but this should rarely, if ever, be necessary.

A NON-SUPPORTED FEATURE

The use of statistics results in a DO expression is not supported in this release of TINY TROLL.

BACKING UP TINY TROLL

It is not possible to make copies of the TINY TROLL disk using commonly available disk copy programs. We have done this in order to prevent unauthorized bootlegging of the program.

You may, however, save the BASIC program called TINY TROLL onto a back-up disk. To do this, simply get into TINY TROLL, then exit via the END command. Insert any other disk into you disk drive and type "SAVE TINY TROLL". This will create a back-up copy of the main program. It is probably a good idea to do this immediately.

In the unlikely event you should physically damage the TINY TROLL disk, you will need to obtain another copy of it from your dealer. If you are a registered user, he or she will be happy to do this for a nominal charge. Be sure to bring the damaged disk with you.

It is a good idea to remove the TINY TROLL disk from the disk drive once you have booted the system. You will not need it thereafter. Keep all data on separate disks.

Dear Customer:

Congratulations on your purchase of TINY TROLL. Please take a few minutes to read this information sheet. It will help you get started as quickly as possible with TINY TROLL.

To run TINY TROLL you should have a 48K Apple computer with a disk drive and Applesoft Basic firmware. This may be an Apple II with Applesoft ROM, an Apple II+, or any Apple with the Language System. To get into TINY TROLL, simply boot your system using the supplied TINY TROLL disk. After a few seconds, the main command menu should appear on the screen.

To help you use TINY TROLL, we have provided extensive documentation. This binder contains a complete reference guide and tutorial manual. In addition, the pages in the inside pocket contain more information you will find useful.

If you have any questions, please contact your Apple dealer or Micro Finance Systems directly. Also, be sure to complete and return the software registration form so you can receive updates and news releases from us.

Thank you.

Micro Finance Systems

TINY TROLL™:

INTERACTIVE DATA ANALYSIS

© 1979 Micro Finance Systems, Inc.
All Rights Reserved

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

Micro Finance Systems makes no warranties, either express or implied, with respect to this manual or to the software described herein. Micro Finance Systems software is sold "as is". In no event will Micro Finance System be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or cosequential damages, so the above limitation or exclusion may not apply to you.

This manual is copyrighted and contains proprietary information. It may not be copied, in whole or part, in any form without prior written consent from Micro Finance Systems.

TABLE OF CONTENTS

PAGE

CHAPTER I INTRODUCTION........................... 1

Overview................................. 1

Brief Historical Digression................. 2

Our Goal................................. 4

Introduction to Time-Series Data Analysis.... 5

Organization of the Manual.................. 9

CHAPTER II THE COMMANDS.......................... 12

Overview................................. 12

Section 1 -- Syntactic Rules................ 12

Section 2 -- A Summary of Tiny Troll's Features........................... 15

Section 3 -- Command Formats................ 24

DEDIT............................... 27

DO.................................. 43

HELP................................ 48

LIST................................ 48

NAMES............................... 50

PR#................................. 54

RANGE............................... 54

SPECIFICATIONS...................... 58

CATALOG............................. 61

LOAD................................ 62

SAVE................................ 63

SLOT................................ 63

STATISTICS.......................... 64

PAGE

PLOT.......................... 66
REGRESSION.......................... 70
CHAPTER III WHY REGRESS?.......................... 79
A non-econometrician's Guide to Econometrics.......................... 80
APPENDIX A DO COMMAND EXPRESSIONS.......................... 90
APPENDIX B ERROR MESAAGES.......................... 92
APPENDIX C SYNTAX SUMMARY.......................... 95
APPENDIX D CUSTOMIZING.......................... 99

TABLE OF TABLES

PAGE

TABLE 1 -- BASIC SYNTACTIC RULES.................. 21

TABLE 2 -- SYSTEM COMMAND SUMMARY................. 22

TABLE 3 -- COMMAND FORMAT SUMMARY................. 25

TABLE 4 -- DEDIT SUB-COMMAND SUMMARY.............. 42

CHAPTER I

INTRODUCTION

OVERVIEW

Apple* Tiny Troll is a new, interactive data analysis language designed to solve problems typically encountered in business, finance, and investment applications. With a simple, single command Tiny Troll can:

- * display high resolution graphs and charts
- * print out tabular reports
- * perform statistical computations on data files
- * transform data in combination with other files and mathematical expressions
- * do estimation and forecasting of time-series data

Because Tiny Troll operates on an entire data file at a time, it is natural to use it for the analysis and investigation of time-series. And because it is a language, you are not limited to a pre-defined set of applications. Some of the power of Tiny Troll can be seen in Chapter 4 where we solve problems in the areas of stock and commodity price charting, financial ratio analysis, estimation of demand function for consumer goods, analysis of securities performance, depreciation calculations, and mathematical function plotting.

Time taken to read through this manual will be well spent. Like anything new, Tiny Troll will take a little while to get used to. Once you do, we're sure you will find it to be a friendly and powerful software tool.

---

* Apple is a registered trademark of the Apple Computer Co.

This manual is a comprehensive guide to Tiny Troll including general descriptions of its capabilities, clear and concise step-by-step instructions in the usage of the specific commands (with appropriate examples), anotated examples utilizing the total package, an instructional section on the purpose and interpretation of multiple linear regression, and useful appendices and tables for quick reference.

A BRIEF HISTORICAL DIGRESSION

Some of you are probably familiar with higher level computer languages like FORTRAN, BASIC, and PL/1. These are general purpose languages, capable of solving a wide variety of business and scientific problems. There is a problem with such generality, however, and that is that the particular application _you_ have in mind may not be particularly easy to implement -- especially if you are not an expert programmer.

Until recently, this was not too severe a problem for most of us because data processing was done in large computing facilities where professional programmers were available to help us. With the advent of microcomputers, though, some of the burden of getting the computer to do what we want it to do has fallen on our shoulders. Additionally, many of you are beginning to use computers for the first time and are finding that the programming languages available (which were developed by and for computer specialists) are difficult, at best, to use.

Economists in the early 1970's had a similar problem. The theory of econometrics, which is the mathematical or quantitative side of economics, had built many powerful theoretical models to explain macroeconomic behavior (e.g., Gross National Product, national consumption) but, in order to use them effectively, computer analysis was necessary. Unfortunately, most economists were not particularly familiar with programming and those who were found existing languages somewhat unwieldy for their purposes.

For that reason, an entirely new language was developed at MIT. It was called TROLL (which stands for Time-shared Reactive On-Line Laboratory), and it enabled the non-computer expert, for the first time, to get the most he or she wanted out of the computer. TROLL is now heavily used in all sorts of econometric and other time-series kinds of analysis. Its utility is apparent and available to even the least computer knowledgeable user because of its interactive nature: you talk to TROLL and it talks to you. Answers come back in seconds instead of hours and TROLL prompts the user for the kind of information needed next. Best of all, it's designed to let the user solve problems in a fashion natural to the problem, and not in a manner artificially imposed by the constraints of the language.

As we have noted, the non-expert microcomputer user faces almost identical problems. In general, here is a wonderful problem solving machine which, without the proper software, is virtually useless.

More specifically, many businessmen and investors want to solve (on a smaller scale) the same kind of problems as the economists: problems of time-series data analysis (sales forecasting, investment decision analysis, etc.). In general, we want to know, given how things were yesterday and the day before, what's likely to happen tomorrow?

OUR GOAL

Our goal was to develop a language with the best features of TROLL which could be run on the Apple II computer. Naturally, we could not even begin to implement the full power of TROLL (which runs on very large IBM equipment), but we did pick a significant subset of its capabilities (e.g., data entry and editing, plotting, modelling, and statistics) that make our version, Tiny Troll, useful.

The commands in Tiny Troll operate on an entire time-series at once. Powerful data transformation can be executed with a single command. Tiny Troll uses the Apple's unique high resolution graphics functions to produce graphs of an exceptional quality and utility. Most important, though, we kept Tiny Troll friendly. As with TROLL, Tiny Troll helps the beginning user to go through it one step at a time until proficiency in its use is gained.

INTRODUCTION TO TIME-SERIES DATA ANALYSIS

Tiny Troll's data files are time-series oriented. A time-series is nothing more than a record over time of a particular event. Each observation of the series tells us what the value of the event was at a particular time in the past. A listing of Mickey Mantle's batting average for every year between 1951 and 1968 would be an example of a time-series.

Associated with each observation or record of an individual event in a time-series is a date indicating when the observation took place. In Tiny Troll, one restriction which we will be limited to is that the events must be evenly spaced in time, or periodic. If we can specify a beginning date and the interval of a time-series, and the number of observations in it, then we will know everything about its structure.

When you create a new data file in Tiny Troll, you will be asked to specify the structure in just such a manner with the exception that you needn't tell it the number of observations in the time-series. Tiny Troll will automatically keep track of the number of observations as you ADD, ERASE, or REPLACE them. You can also change these SPECIFICATIONS after you have created the file, if you wish. But why do we use a time-series? What is the advantage of such a procedure?

Most practical forms of research, be they in business, finance, or social science, are really attempts at forecasting the future: What will the price of a stock be? What will the sales of a particular item be over a certain period of time? How will the public react to a new social program? Since most of us don't have a crystal ball with which to look into the future, a good way is to look into the past as an aid in understanding the future. Thus, we want to consider instances of what has (historically) been the case. The methods of statistics and data analysis, in general, are used to help us make sense out of the history of events.

While we can't easily give a number to many very important things in life (e.g., the quality of life), there are many important events which we can quantify (e.g., Dow Jones averages, Gross National Product). These quantifiable events are then analyzed in order to predict, or get a feel, for the future. Time-series, therefore, is the most efficient method of organizing historical data for the analyses necessary to predict likely future trends, and that's why we use it in Tiny Troll.

There are several methods in common use for the analysis of time-series data. One method, of course, is to simply look at a string of numbers and intuitively determine their meaning. For example, comparing the annual amounts of snow in Boston and Miami over the past twenty years would not demand highly sophisticated statistical techniques to determine that it never snows in Florida while it snows rather regularly in Boston. Tiny Troll's LIST command facilities provide this kind of simple comparison.

We can learn some more specific things by computing some simple statistical functions on a series. The mean tells us how much snow fell in Boston, on the average. Was it one inch or eight feet? Another measure, the variance, tells us about the spread of amounts of snowfall. Did the same amount of snow fall every year or were some years much heavier than others? A low variance indicates that most of the observations were clustered around the mean and that there wasn't much variation from time to time. A high variance, on the other hand, suggests that there was a great deal of variation in the amount of snowfall. This would be important information to have if we wanted to predict next year's snowfall amount. The STATISTICS command in Tiny Troll provides this information as well as the maximum and minimum observation and the series' standard deviation (the square root of the variance which is used in many more advanced statistical methods).

Often, as the saying goes, a picture is worth a thousand words or, in this case, numbers. A pictorial representation of one or more time-series can be very useful n uncovering trends within the series. The essence of PLOTting is that it draws a picture, over time, of an event. Is it generally increasing? Decreasing? Oscillating? Are there peaks? How often do they occur? Is the series' behavior regular or irregular? All of these questions can be investigated via a PLOT. We believe the PLOTting package in Tiny Troll is the finest graphics package available for microcomputers. You can examine files collectively or singularly, with the axes of the graph being labelled and scaled automatically.

Finally, more advanced methods of data analysis use the techniques of estimation and modelling. That is, one can mathematically construct a theory of the relation between a variety of kinds of observations and one of particular interest. To what degree can we predict the observation of interest from the other observations? We might, for example, think that there is a relationship between the state of the economy (i.e. income levels, national output, unemployment) and the stock market. Given that we know the historical values of all of the variables, we can use the technique of multiple linear regression to tell us whether there really is a connection and how good the predictive quality of that connection is.

Using Tiny Troll we can construct a model which attempts to explain stock market prices in this fashion. One uses multiple linear regression to estimate the coefficients of the model to tell us how important, relatively speaking, each of the independent variables are in explaining the behavior of the dependent variable, and, also, to see how good a prediction could be made of the dependent variable using the coefficients of the independent variables. Then we can attempt to do some forecasting. If we make some assumptions about the future values of the independent variables, we can use them to predict future values of the dependent variable.

This process of building a model and then forecasting with the model, is one of the most powerful tools of data analysis. It plays important roles in business, in finance, in social science research, and in general policy and decision making. Tiny Troll is unique in having a sophisticated modelling and forecasting capability for microprocessor based systems, thus making the power of these functions available, for the first time, to a wide variety of users.

ORGANIZATION OF THE MANUAL

It is highly suggested that all Tiny Troll users, experienced programmers and novices alike, read this instructional manual carefully. As noted earlier, Tiny Troll *is* a language and, like all languages, takes some getting used to. This manual is organized in a fashion designed to make the getting acquainted process rapid and painless, by progressively helping you to develop an understanding of the commands and applications of Tiny Troll.

The first chapter, which you have already read, contains an elementary history of, and introduction to, Tiny Troll and time-series data analysis. The second chapter covers the capabilities and syntactic rules for the various Tiny Troll commands, and provides step-by-step examples of the implementation of the individual commands.

Chapter 3 provides a concise, but comprehensive, introduction to multiple linear regression and modelling, and explains the meanings of the output of Tiny Troll's REGRESSION command.

Anotated examples supplied on the Tiny Troll demonstration diskette, provide you with a variety of real-life situations to which Tiny Troll may be applied (e.g., security valuation, sales forecasting)

Finally, the appendices allow you to reference command summaries, error messages, and other information rapidly so that you can answer questions, which arise during Tiny Troll sessions, quickly and easily.

Constructive criticism about this manual is welcome at any time. We are especially interested in suggestions you may have which might make learning about and using Tiny Troll simpler. Additionally, while Tiny Troll has been carefully tested prior to release, you may discover "bugs" in the program which have not been previously noted. Please don't hesitate to contact us, in such an event, so that we can correct the problem and so help you to utilize the full power of Tiny Troll.

CHAPTER II

THE COMMANDS

OVERVIEW

The purpose of this chapter is to guide the new user step-by-step through the various features and functions of Tiny Troll and to provide him with a variety of examples for each. The chapter is divided into three sections. The first section orients the reader to the overall format of Tiny Troll by explaining syntactic and procedural rules which are general to the language, the second section supplies a summary of the capabilities of Tiny Troll, and the third section provides specific information on each of the functions of Tiny Troll.

SECTION 1 -- SYNTACTIC RULES

In this section we will discuss some basic things that are general to all aspects of the utilization of Tiny Troll, and which are necessary for the user to know before implementing the program. These include how to start the system, command formats, editing procedures, prompts, and error messages.

Tiny Troll is designed to be run on the Apple II computer with 48K of RAM, Applesoft in ROM, and at least one (1) disk drive. If hard copy is desired, a 40, 80, or 132 column printer (parallel or serial) can be used. For most hard copy applications an 80 column or more printer is desirable.

In order to load Tiny Troll properly it is important to boot the diskette each time it is loaded. This is accomplished by inserting the diskette in the disk drive and then booting the diskette in the usual manner. If you are uncertain how to do this, consult your DOS 3.2 manual for the precise instructions. The reason for booting the diskette each time Tiny Troll is loaded is that a utility program is run during the booting process.

After successful booting of Tiny Troll you will see the message:

```
*** TINY TROLL     V1.0 ***
© 1979  MICRO FINANCE SYSTEMS
```

followed by the command menu and the prompt:

```
COMMAND>
```

The program diskette can now be removed from the drive and a diskette containing data files, or a blank diskette for storing new data files, can be inserted. Tiny Troll is now ready for work, and you are ready to learn to work with it.

Tiny Troll is a highly interactive language which affords the user a high degree of flexibility. Commands can be entered in a variety of sequences and lengths. You may enter part of a command, an entire command, or a series of commands at once. After you enter each line, Tiny Troll responds with an informative prompt indicating the type of information required next. By entering shorter lines the user, in effect, asks for guidance from Tiny Troll and receives it. The beginning user will find this feature particularly valuable since he can simply enter the words in a command one at a time (with a return after each). This is an excellent way to teach yourself the command structure. This is true for all the commands in the command menu except for PLOT and REGRESSION, as will be discussed later in this chapter. The experienced user can simply enter an entire command, or series of commands, at once, if this is convenient.

A command is constituted by a keyword (e.g. LOAD, NAME, DO, PLOT) followed by a series of arguments (filenames, data specifications and others we will learn about) and terminated with a colon (:). While there are a few exceptions to the rule that the keyword must be followed by a filename (e.g., HELP, CATALOG, PR#, NAME), most commands related to data entry, analysis, or modification do adhere to the above format.

There are only two basic syntactic rules which are general to the entry of all commands in Tiny Troll and which will produce error messages if not followed explicitly. First, it is important that each word or number in an entry be separated by at least one space since Tiny Troll recognizes each part of a command by the empty spaces surrounding it. The second basic rule is that each completed command must end with a space followed by a command terminating colon (:). Remember, if you are only entering part of a command on a line, don't use the colon but simply hit return.

(see additional documentation, p. A1)

Examples will make this clear.

If you make a mistake prior to hitting return at the end of a line, the editing procedure is exactly the same as in Applesoft BASIC: simply use the back and forward arrows in the usual manner to correct your errors. You may also use the control-x key to cancel an entire line, as in BASIC. If you should happen to hit return at the end of a line with an error in it, Tiny Troll will respond with an error message specifying the type of error and re-prompting you for the next type of input required. Notice that when you do make a mistake, Tiny Troll recovers without losing any of the information which you previously entered. Appenix B lists the possible error messages you can get and their meanings.

SECTION 2 -- A SUMMARY OF TINY TROLL'S FEATURES

This section provides a summary of the individual functions and features of Tiny Troll; section 3 should be consulted for the specific capabilities, syntactic formats, limitations, and examples of each function.

Tiny Troll provides simplified data file entry and editing capabilities which allow you to enter and modify numeric data files easily. A data file in Tiny Troll is essentially a linear list or series of data values. The data of each series is stored in a separate disk file. Since Tiny Troll has its origins in econometric, time-series research, each data entry will be automatically associated with a particular date. This does not pose any limitations, however, in utilizing data which is not time associated, as will be discussed below.

The commands related to file entry and modification include DEDIT, LOAD, NAME(S), LIST, RANGE, SPECIFICATIONS, DO, SAVE, SLOT, and DRIVE.

DEDIT (for Data EDIT) and the associated sub-commands ADD, ERASE, REPLACE, LIST, and QUIT, allow for the entry of new data files from the keyboard and for the modification of files already active in the system. All files are identified by a one word name of eight characters or less not beginning with a number. You can enter a maximum of ten files into Tiny Troll with a maximum of one hundred data points in each file. Both of these values, however, can be altered.

During the new file definition process, the year and period of the year to be associated with the start of data entry is specified. In the case of non-time oriented data, a starting year of 1 and periodicity of 1 is selected.

LOAD allows for the LOADing of files into Tiny Troll from data files stored on diskette.

The command NAME(S) has two forms; one which provides a listing of the filenames currently active in the system, and one which provides the same listing with the additional capacity to change the names of the individual files. Each filename listed will have a filenumber associated with it which will be used instead of the filename in a few commands.

The LIST feature allows for the listing, in tabular form, of the contents of one or more active files. The listing comes in two phases. First a "header" is displayed with the name of the file being LISTed, the range of the file, the start and end dates, and the number of cases in the file. Second the data in the file is listed with the filename used as the heading for the column. During the LISTing process, a flashing cursor indicates that any key can be pressed in order to continue the LISTing. (Naturally, reset and control-C are not acceptable keys to press because they will halt the running of Tiny Troll.) The LIST feature is particularly handy in the generation of reports in tabular forms.

RANGE and SPECIFICATIONS both allow the user to modify existing data files. RANGE affords you the ability to choose a sub-set of a particular data file and SPECIFICATIONS enables you to modify the dates associated with the data in the file.

The DO function, perhaps the most powerful Tiny Troll function related to file development, allows for the transformation of the data in a file according to a mathematical expression and either the creation of a new file with the transformed data or the replacement of the original data with the transformed data under the original filename. Thus, roots, logarithms, powers, logical transformations and statistical transformations of entire files may be executed in a single command.

The SAVE feature allows you to SAVE on diskette any file at any time. It is highly suggested that the user SAVE any new files entered into the system, or newly transformed files, immediately after entrance or transformation in order to safeguard the files.

The SLOT and DRIVE commands allow those users with two or more disk drives to select which drive they want to utilize for CATALOGing a diskette or for SAVing or LOADing a file.

The preceding commands give you great flexibility in entering, modifying, LISTing, and SAVing data files, now you need commands which will enable you to utilize those files in order to provide yourself with answers to your research questions. The following commands will help you in this process.

The STATISTICS command supplies you with the basic statistical properties of any active file. Along with displaying the header of the file, this command computes the maximum, minimum, mean, variance, and standard deviation of the file. In addition, STATISTICS computes, but does not display, the sum and the sum of squares of the file. These two measures, along with the maximum and minimum of the file, are stored for later use in the right hand argument of the DO command.

The PLOT command provides you with a particularly powerful visual tool for understanding the relationship between the data in two or more (maximum of six) active data files of equal range and periodicity or for seeing the fluctuations within the data of one file. The PLOT command employs the Apple II's high resolution graphics with a number of important features not available in other microcomputer software packages. Among these features are automatic axis labelling, scaling of data, and a plot legend.

The X axis is used by Tiny Troll to represent the year and period associated with the data in the file and the Y axis is used to represent the value of the data itself. The scaling on the X axis is automatically calibrated according to the dates of the data in the file and the scaling on the Y axis is likewise calibrated according to the range of the data values to be represented. If appropriate, Tiny Troll will adjust the Y scale according to the exponential value of the data so that the numberical labels of the Y axis will not exceed four digits. The exponential value employed is then listed above the Y axis in the upper left corner of the screen. Each individual data file is also assigned a distinctive symbol for easy identification. These symbols are then listed in a legend below the graph.

If hard copy of a PLOT is desired, a special graphics type printer, such as the Integral Data or Axion, must be used.

The REGRESSION command provides you with the capacity to perform multiple linear regression with as many as nine independent variables utilizing the ordinary least squares estimation. This command is a valuable tool in model development and in helping you to forecast future trends. This is particularly useful in financial analysis, marketing forecasting, and social science research. The output of the REGRESSION command yields estimates of the coefficients of the variables and a variety of statistics including measures of the goodness of fit, the colinearity, the correlation, the variance-covariance interactions, and the standard errors of the REGRESSION. Tiny Troll also computes fitted values for the dependent variable from the coefficients of the independent variables which are produced by the REGRESSION and calculates the differences between the predicted and the actual values, the residuals, and allows you to store the residuals for future use or PLOTting. (For an in-depth discussion on the purposes and the pitfalls of REGRESSION, please refer to Chapter 3, "Why Regress?".)

There are two commands, CATALOG and HELP, which are provided solely as informational aids to Tiny Troll users. CATALOG displays the directory contents of any diskette at any point during a Tiny Troll session. This feature helps you to search for specific files which can then be LOADed into the system. The HELP command is just that: HELP! If you get stuck and don't know what to do next, at any time when the system is ready to receive a command, typing HELP displays the commmand menu from which you can select your next command. If you already know what command you want to use but are not sure of the appropriate procedure for entering that command, just type the command name (DEDIT, DO, etc.) and Tiny Troll will prompt you for the type of information needed next. This is true, as we shall see, for all the commands except PLOT and REGRESSION.

The last command in Tiny Troll's command menu is PR# which allows you to turn your printer on and off. This command is capable of being executed not only when a command prompt is displayed but, also, when an edit command prompt is displayed in DEDIT mode.

TABLE 1

BASIC SYNTACTIC RULES FOR TINY TROLL

RULE 1:	Each word in a command must be separated by a space since Tiny Troll recognizes each part of a command by the empty spaces surrounding it.
RULE 2:	Each complete command must end with a space followed by a colon. This is true even if you place more than one command on a line.

(see additional documentation, p. A1)

TABLE 2

TINY TROLL SYSTEM COMMAND SUMMARY

SYSTEM COMMAND		DESCRIPTION
SUPPORT COMMANDS	DEDIT	Allows for the entering of new data files from terminal, specifying new file name and header, and ADDing, REPLACing, or ERASing data in existing files.
	DO	Transforms data files according to a mathematical expression and stores the transformed file under the original filename or a new filename.
	HELP	Displays command menu at any point during a Tiny Troll session.
	LIST	LISTs, in tabular form, the elements of one or more data files. Particularly useful for the generation of reports.
	NAME(S)	Two forms: One allows for the displaying of the NAMES of the active files in Tiny Troll and their associated filenumbers; the other provides the same display plus the option to change the NAME of any file.
	PR#	Turns the printer on and off. This can be accomplished in regular command mode or in the DEDIT mode.
	RANGE	Allows for the selection of a subset of consecutive entries from a data file. Temporary.
	SPECIFI-CATIONS	Allows for the alteration of the header SPECIFICATIONS (start year, start period, periodicity) of a data file. Permanent.
DISK COMMANDS	CATALOG	Displays the directory contents of diskettes in order to allow for location and LOADing of data files.
	LOAD	LOADs data files from diskette into active Tiny Troll file locations.

SYSTEM COMMAND		DESCRIPTION
DISK COMMANDS	SAVE	SAVEs data files from Tiny Troll to diskette. Should be done immediately after file has been created or transformed in order to safeguard file.
	SLOT DRIVE	For systems with two or more disk drives, allows for the selection of SLOT number and DRIVE number for SAVing, LOADing, or CATALOGing.
ANALYTIC COMMANDS	PLOT	High resolution graphing of the elements in one or more data files with automatic axis labelling, scaling of data, and PLOT labelling.
	REGRESSION	Allows for multiple linear regression of one dependent variable with up to nine independent variables (can be increased) using the ordinary least squares estimation method. Estimates the coefficients of the variables and provides a variety of statistics.
	STATISTICS	Displays basic statistical parameters of a data file (number of cases, maximum, minimum, mean, standard deviation, variance) and stores maximum, minimum, sum, and sum of squares for use with DO command.

SECTION 3 -- COMMAND FORMATS

By now you should have a fairly good idea of what Tiny Troll has to offer. This section is designed to give you the specifics necessary to use all the features of Tiny Troll. Each command will be completely described in terms of its capabilities, the syntax of its usage, and its limitations. Clarifying examples are liberally used throughout the section.

There are some basic conventions which are used in representing the syntax of command statements which you need to know. Essentially, an entire command is of the form:

```
KEYWORD argument [argument...] :
```

A KEYWORD is a word that must be entered with a fixed spelling (or two letter abbreviation) that Tiny Troll is programmed to recognize. Each argument (such as a filename or a numeric value) is an item needed to complete the meaning of a command. The KEYWORD must always be copied just as it is in the command menu. The lower-cased arguments, however, are to be replaced by an appropriate filename, data value, or date. Remember, to signify the end of a command you must leave a space after the final argument and then type a colon (:). This does not mean that every line you type must have a colon since you can enter most command KEYWORDs and arguments one at a time. Use a colon <u>only</u> when you come to the end of an entire command; this, as we shall see, can happen more than once on a line.

In the standard form represented above, the brackets ([]) are indicative of an optional argument or arguments and the elipses (...) indicate that the optional argument may be entered repeatedly in one command.

It is important to re-emphasize that each word (KEYWORD or argument) in a command must be separated by a blank space. This is crucial.

TABLE 3

COMMAND FORMAT SUMMARY

---

1 -- Every command begins with a KEYWORD which may be abbreviated to its first two letters (e.g. PL, NA).

2 -- Every complete command must end with a colon (:).

3 -- You must enter a blank space between each word of a command.

(see additional documentation, p. A1)

BOOTING TINY TROLL

As mentioned previously, in order to utilize Tiny Troll you need an Apple II computer with 48K of memory in RAM, Applesoft in ROM, and at least one (1) disk drive. Booting is necessary prior to each Tiny Troll session; just running the program on a DOS which has already been previously booted will not work. If you have any questions as to the proper procedure for booting the Disk Operating System, refer to your DOS 3.2 manual.

After successful booting of Tiny Troll, the following will appear on you CRT:

```
                  *** TINY TROLL    V1.0 ***
                © 1979  MICRO FINANCE SYSTEMS


     CA -- DISK CATALOG
     DE -- DATA EDIT
     DO -- DO DATA TRANSFORMATION
     HE -- HELP
     LI -- LIST
     LO -- LOAD
     NA -- SET FILENAMES
     PL -- PLOT
     PR# -- SET PRINTER
     RA -- SET ACTIVE RANGE
     RE -- REGRESSION
     SA -- SAVE FILE
     SP -- SPECIFICATIONS
     ST -- STATISTICS


COMMAND>  (followed by flashing prompt cursor)
```

If this does not happen, boot the diskette again. If the above heading, command menu, and command prompt do appear, you can remove the Tiny Troll program diskette and replace it with another diskette which will be used for SAVing and LOADing data files. Now both you and Tiny Troll are ready for work.

The following instructions contain explanatory examples and it's highly suggested that you now (if you have not already) boot Tiny Troll and implement the examples as they are supplied.

DEDIT

SYNTAX: DEDIT [NEW] filename [periodicity start-year start-period] :

ADD date value [value...] :

REPLACE date value [value...] :

ERASE date :

ERASE date date : (for ERASing all the data values between two dates)

LIST :

PR#n : (where n is the number of the printer slot)

QUIT :

DESCRIPTION:

The DEDIT command, and its attendant sub-commands (ADD, REPLACE, ERASE, LIST, PR#, and QUIT), allow you to enter new data files from the keyboard and also to alter existing files in a variety of ways.

Let us start using this command by entering a new data file named "HOWDY" (filenames must start with a letter, must not have any embedded blanks, and have eight or less characters (letters, numbers, or special characters)).

Tiny Troll indicates that it is ready to accept a command by displaying a flashing cursor after the command prompt:

```
COMMAND>
```

Next, follow the command prompt with the word DEDIT:

```
COMMAND>  DEDIT
```

followed by hitting return. Because Tiny Troll is interactive and prompts you along the way, Tiny Troll will respond with a prompt asking for more information. In this case Tiny Troll responds with a request for the name of the file you want to edit:

```
FILENAME:
```

If you wanted to edit a file which is already active in the system (i.e., it existed on diskette from a previous Tiny Troll session and you LOADed it or generated it previously in this session), you would only have to type in the name of that file. In this case, you want to let Tiny Troll know that you are creating a new file. This is done by typing in the word NEW after the last prompt:

```
FILENAME: NEW
```

and now hit return. Tiny Troll will respond with another request for a filename. This does not mean that Tiny Troll didn't understand your NEW command but indicates that Tiny Troll is now ready to record the new file's name, so type in "HOWDY", and hit return:

```
FILENAME: HOWDY
```

Tiny Troll has now opened a new data file for you with the name "HOWDY" and it needs some more information from you about "HOWDY" before you can start editing data into it. It tells you this by responding to your last line with:

```
PERIODICITY:
```

As noted in Chapter I, Tiny Troll has its origins in time-series econometric research, so Tiny Troll automatically assigns dates to the individual members of data files. To do this, it first needs to know what segments of the year the individual pieces of data being entered into the file will represent. This is known as the file's "periodicity". A periodicity of "4" means that data will be quarterly. You can assign any periodicity you would like but the most common, in addition to quarterly data, are "1" for annual data, "12" for monthly data, "52" for weekly data, and "365" for daily data. Let us say that the data you want to enter into "HOWDY" is monthly, so type in "12" after the last request from Tiny Troll:

```
PERIODICITY: 12
```

and hit return. Tiny Troll responds with:

```
START YEAR:
```

Tiny Troll is now requesting to know what year your monthly data is to begin with. How about 1975? It is recommended that, when possible, the years associated with the data in your files be represented by a two digit number rather than the usual four digit number. This is done in order to save space in the label of the X axis when data is PLOTted. In this case, "1975" can be entered as "75" without losing any important information. This convention will be used throughout this manual.

```
START YEAR: 75
```

Don't forget to hit return. Now Tiny Troll responds with:

```
START PERIOD:
```

Tiny Troll wants to know what period (in this case, which month) in "75" you want to start entering data. Let's use period "3" (March). Type in a "3" and hit return:

```
START PERIOD: 3
```

Tiny Troll should now give you an interesting response that looks something like this:

```
HOWDY
PERIOD 12    START DATE  75:3     CASES=0
RANGE  75:3  TO  75:3  WITH 0 CASES
EDIT COMMAND>
```

The information before the "EDIT COMMAND" prompt is known as the "file header". It will appear whenever you LIST the file and will carry the file's vital statistics. Let's take a look at it: On the first line it has the file's name, "HOWDY". Then it tells you, on the second line, the file's start date and period. All of this you entered into Tiny Troll when you defined the file. The header also gives you two other pieces of information which change automatically as you ADD, REPLACE, or ERASE data from the files. "CASES" let's you know how many data entries currently exist in the file. This will be 0 now, since you haven't entered any data into "HOWDY" yet. In the third line of the header, "RANGE" tells you the start year and period and the end year and period of the currently active subset of the file. (This will be explained when we discuss "RANGE" below.) For "HOWDY", at the present moment, the start and end of the "RANGE" are the same -- "HOWDY" is an empty file.

Once you are more familiar with the command structure, you can enter all or some of the above command entries on one line as a single command:

```
COMMAND>  DEDIT NEW HOWDY 12 75 3 :
```

This data filing system is, as noted, time-series oriented. If your data does not represent time asscciated information, you may find that a periodicity of 1, a start year of 1, and a start period of 1 is a useful way to label the file. Whenever a file with a "PERIODICITY" of 1 is either LISTed or PLOTted, the period will not be displayed so it will look like ordinary non-time-series data.

The edit command prompt which follows the header provides you with the opportunity to ADD, REPLACE, ERASE, or LIST the data in your data file, set the printer, and QUIT the DEDIT mode. In the case of "HOWDY", you can only ADD data since it is an empty file. You can proceed with data entry by typing in, after the edit command prompt, the word ADD followed by a return:

```
EDIT COMMAND>  ADD
```

Tiny Troll then responds with:

```
YEAR:PERIOD
```

This is a request for the year and period with which you want to start entering data. Since you want to start by placing data at the beginning of "HOWDY", type in the year and period of the beginning of the file without any spaces, as follows:

```
YEAR:PERIOD : 75:3
```

but, in this situation, don't follow the entry with a return; you have to follow this entry with the actual data entries -- Tiny Troll expects it. You can type in up to twenty data entries on a line. Tiny Troll will automatically associate the first data entry with the year and period you previously entered on the line and each successive data entry with the next available year and period. Let us say you want to make ten monthly entries (35, 22, 50, 42, 8, 12, 15, 40, 55, 30) starting with 75:3, then type:

```
YEAR:PERIOD : 75:3 35 22 50 42 8 12 15 40 55 30
```

If this is all you want to enter at this time, then follow the last entry with a space (don't forget the other spaces) and a colon. Now hit return. (You will no longer be told to hit return each you want to enter a line; it will now be assumed that this will be an automatic reponse on your part.) Tiny Troll's response is:

```
EDIT COMMAND>
```

Tiny Troll has now associated 35 with 75:3, 22 with 75:4, 50 with 75:5, etc. If you want to see this LISTing, all you have to do is type LIST after the edit command prompt:

```
EDIT COMMAND>  LIST :
```

You will now see the header of "HOWDY":

```
HOWDY
PERIOD 12    START DATE  75:3     CASES=10
RANGE 75:3   TO  75:12  WITH 10 CASES
```

As you can see, the header was automatically updated as we mentioned earler and it is followed by a flashing cursor which indicates that Tiny Troll is waiting for you to let it know that you want to see the rest of the LISTing. This is accomplished by hitting any key (except reset or control-C). The contents of "HOWDY" will now be displayed for you:

```
DATE          HOWDY
----          -----
75:3          35
75:4          22
75:5          50
75:6          42
75:7          8
75:8          12
75:9          15
75:10         40
75:11         55
75:12         30
```

You can use the LIST command after any edit command prompt in the DEDIT mode and the contents of the file being edited will be displayed.

If you want to enter more than the allotted twenty data entries per line (of course, you can enter as few as one cr none -- just type a colon), or if you want to know at what point you are in the data file, you should substitute the colon with a semi-colon (;):

```
YEAR:PERIOD : 75:3 35 22 50 42 8 12 15 40 55 30 ;
```

to which Tiny Troll will respond:

```
76:1  NUMERIC VALUE:
```

This is Tiny Troll's way of letting you know that the next available file location in "HOWDY" is 76:1 and that it is ready to associate the next data entry with that year and period. Again, you can enter up to twenty data values on the line and you can terminate the data entry process with a colon or let Tiny Troll know that you want to continue data entry by typing a semi-colon at the end of the line.

There is an easier way to enter the initial data entries into a file. You don't have to specify the start date and period of the file but, instead, you can simply enter the word TOP after ADD and follow it with your data entries, thus:

```
EDIT COMMAND>  ADD TOP 35 22 50 42 8 12 15 40 55 30 :
```

will have the same effect on "HOWDY" as the edit command which specified the start year and period.

The TOP (and similar BOTTOM) command can be used anywhere in DEDIT instead of a year and period when data is to be ADDed, REPLACEd, or ERASEd at the beginning (or end) of a file.

Each time you complete an edit command, Tiny Troll will respond with another edit command prompt. In order to exit from the DEDIT mode, all you need do is follow the edit command prompt with the word QUIT. Tiny Troll will then respond with the command prompt indicating that it can now be commanded to perform any of the functions available in the command menu, and that the file you were editing is being held in Tiny Troll's active file subset for future use.

So far we have been talking fairly specifically about the creation of, and data ADDitions to, new data files. Files already active in the system can be altered in a similar manner. You can ADD data, in the same fashion, to the TOP, middle, or BOTTOM of a file with previous entries. In addition, there are two other subcommands (REPLACE and ERASE) in the DEDIT mode which are specifically designed to alter the data in existing files.

"HOWDY", as noted, is now considered to be a previously defined file which is being stored in Tiny Troll's active file subset. Let's see how we can alter "HOWDY", or any previously defined file, using the DEDIT command.

First, let's type the following:

```
COMMAND>  DEDIT HOWDY :
```

This command tells Tiny Troll to bring "HOWDY" from the active file subset into the DEDIT mode and Tiny Troll's response:

```
EDIT COMMAND>
```

let's you know that it is ready to edit the file. Since you may not remember exactly what the contents of "HOWDY" are, you may want to first type:

```
EDIT COMMAND>  LIST :
```

in order to display the header and contents of "HOWDY".

Now, let's use the REPLACE command to substitute some of the old data in "HOWDY" with new data; again, Tiny Troll will prompt you through the process. Let's REPLACE the value of 8 which is associated with 75:7 with the value 20. To do this, type:

```
EDIT COMMANC>  REPLACE
```

to which Tiny Troll responds:

```
YEAR:PERIOD :
```

which is a request for the date and period of the value to be REPLACEd, so you 75:7 followed by the new value:

```
YEAR:PERIOD : 75:7 20 :
```

Tiny Troll now gives you a new edit command prompt. To make sure that the data was successfully REPLACEd, LIST the contents of "HOWDY" once more. Voila!! Instead of finding 8 opposite 75:7 we find 20, our new value. This command sequence can be shortened by the experienced user to:

```
EDIT COMMAND>  REPLACE 75:7 20 : LIST :
```

Notice that you can place more than one command on a line and Tiny Troll won't mind.

You can REPLACE up to twenty data values on one line; they will then REPLACE the data in the consecutive data locations after the first one specified. Typing QUIT will get you out of the DEDIT mode when you are finished editing your file.

The ERASE command allows you to remove the value of a specific data location from an existing file. It should be noted that Tiny Troll does not substitute the value with a zero or leave it blank, but, rather, moves all of the data locations which follow the one which was ERASEd up one location. For example:

```
COMMAND>  DEDIT HOWDY
EDIT COMMAND>  ERASE 75:7 :
EDIT COMMAND>  LIST :
```

doesn't simply leave 75:7 empty but moves values up to fill the gap. Similarly, ADDing a value at the TOP or middle of a file pushes the value already in the specified data location, and all the subsequent data values, are pushed down one data location. If two or more new values are ADDed to the TOP or middle of a file the old values are pushed down by the number of new values. ADDing to the BOTTOM simply creates an appendage to an existing file. It is recommended that the ERASE command be used primarily to correct errors such as the entry of erroneous data. It should also be noted that this command should not be used to delete the whole file. You can, however, ERASE a number of locations on the same command line:

```
EDIT COMMAND>  ERASE 75:7 75:8 75:9 ... :
```

If you desire to ERASE only one data value you should do it all on one line, as a complete command. Otherwise, if you want Tiny Troll to prompt you through the procedure, Tiny Troll will expect you to ERASE two values at a time and it will be necessary for you to either follow the date of the data location to be ERASEd with a colon or to enter the date twice:

```
EDIT COMMAND>  ERASE
YEAR:PERIOD : 75:7
YEAR:PERIOD : 75:7
EDIT COMMAND>
```

As with the REPLACE command, the experienced user can combine the ERASE commands into one line:

```
COMMAND>  DEDIT HOWDY : ERASE 75:7 75:8 : LIST :
```

ERASE also allows you to remove a range of data values by indicating, in one command line, the first and the last data locations whose values you want to ERASE.

```
EDIT COMMAND>  ERASE 72:3 74:2 :
```

The above command provides for the deletion of _all_ data values in locations 72:3 through 74:2, inclusive, and pushes all subsequent values up in order to fill the gap.

If you want to ERASE the values in two data locations which are _not_ consecutive, you should _not_ place them both on the same line or you will also, inadvertently, ERASE all the intervening data location values.

Again, QUIT will get the command prompt back.

One last feature of the DEDIT command is that you can turn the printer on and off by following any edit command prompt with a PR#1 to turn the printer on (another number can be used if your printer is in a different slot) or a PR#0 to turn it off.

LIMITATIONS:

1. Unless modified, you can only enter one hundred data values into a data file. (See Appendix D.)

2. Unless modified, you can only have ten data files active in the system at one time. (See Appendix D.) If you do enter more than ten files, Tiny Troll will replace the first file with the eleventh, the second with the twelfth, etc. and give you no indication that it has done such. You should occasionally check the number of active files you have, by utilizing the NAMES command, in order to avoid losing important information.

3. You may only DEDIT one file at a time.

4. If you make a syntax error during the DEDIT procedure and have not corrected it prior to hitting return, you must re-enter the whole command.

5. When a data value is ERASEd the following data values in the file do not retain their positions but move up one data location.

6. DEDIT does not provide for the deletion of a complete data file from the active file subset. ERASE cannot be used for this purpose. If you want to delete an entire file, you should SAVE the files you want to SAVE onto a diskette, re-boot Tiny Troll, and the re-LOAD the files you want into the system.

TABLE 4

DEDIT SUB-COMMAND SUMMARY

SUB-COMMAND	DESCRIPTION
ADD	ADDs data entries to the beginning, middle, or end of a data file. Pushes data in file down.
BOTTOM	Can be used instead of date to ADD, ERASE, or REPLACE data at the BOTTOM of a file.
ERASE	ERASES data from specific data locations, or from a consecutive series of data locations. Pushes remaining data up.
LIST	LISTs header and contents of file currently in DEDIT mode.
NEW	Allows for creation of new data files.
PR#	Provides for turning printer on and off from within DEDIT mode.
QUIT	Allows exiting from DEDIT mode.
REPLACE	REPLACEs data in specific data locations. Does not effect other data in file.
TOP	Can be used instead of date to ADD, ERASE, or REPLACE data at the TOP of a file.

DO

SYNTAX: DO filename = expression :

DESCRIPTION:

The DO command allows for the creation of a file by transforming the data values in an existing file or files according to an arithmetic, trigonometric, or logical expression you specify.

As an example, create a file, "TOMATO", which has data values 15, 22, 43, 10, and 8, with a periodicity of 1, a start year of 1, and a start period of 1. Use the DEDIT function for accomplishing this and, when you are finished, get the command prompt back. Now let's create a new file, "ONION", using the DO command by dividing each data value in "TOMATO" by 2. This done by first typing:

```
COMMAND>  DO
```

Tiny Troll will respond to this with:

```
TRANSFORM:
```

which is a request for a number of pieces of information. First Tiny Troll wants to know the name of the file that you want to store the transformed data in. You can either store the data under the same filename you are getting the original data from (thus REPLACing the original data with the transformed data) or you can use a new filename. Remember to make certain that you are not creating an eleventh, twelfth, or thirteenth file, thus losing the data from the first, second, and third files. In the case where you are creating a new file, Tiny Troll opens a file with the same header as that of the file being transformed. In this example, you would type:

```
TRANSFORM: ONION
```

If you hit return after entering "ONION", Tiny Troll would respond with a syntax error statement which, in this case, would indicate that just typing "ONION" wasn't enough. Tiny Troll also wants to know, on the same line, the name of the file(or files) you want to transform, and the expression by which you would like to transform it (or them). The filename(s) must be part of the expression. Any arithmetic, trigonometric, or logical operation which is legal in Applesoft BASIC is legal in Tiny Troll; this also includes logarithmic functions, RND, ABS, and EXP. (See Appendix A for a list of the expressions you can utilize with the DO command.) In our example, the expression is "TOMATO/2". The command line now looks like this:

```
TRANSFORM: ONION TOMATO /2 :
```

An optional equal sign (=) may be included for clarification purposes:

```
TRANSFORM: ONION = TOMATO /2 :
```

This whole command could also be written in one line:

```
COMMAND>  DO ONION = TOMATO /2 :
```

These commands are identical and create a new file, "ONION", with values 7.5, 11, 21.5, 5, and 4. LIST file "ONION" to make sure.

Notice that it is necessary to leave empty spaces around all the filenames in the DO command -- you _cannot_ eliminate the spaces in the expression. You may also include two or more active filenames on the right hand side of the DO command. For example, to create a new file, "SUM", in which each element is the sum of the two corresponding values of the original files "TOMATO" and "ONION", you would say:

```
COMMAND>  DO SUM = TOMATO + ONION :
```

The header for the new file would, in this case, be the same as the header for "ONION" -- the periodicity, start date, start period, and number of cases would be identical. In general, when an expression contains more than one active filename, all of the files must have the same periodicity and number of cases, and the new file's header will be copied from the _last_ active file specified in the expression. If you wish to transform two files of unequal sizes, set the RANGE of the larger file to be equal to that of the smaller file.

An expression can be of arbitrary complexity subject to the one limitation that it contain less than twenty blanks. (Contrary to the general rule, except for filenames, parts of an expression need not be separated by spaces.)

There are a few additional facilities which are available to the DO command as a result of the STATISTICS command. If one or more STATISTICS commands (on different files) are performed first, you can use the following arguments in the right hand expression of a DO command:

SUM(i) -- The sum of the data points in file "i".

MAX(i) -- The maximum data value in file "i".

MIN(i) -- The minimum data value in file "i".

SS(i) -- The sum of squares of the data points in file "i".

The "i" represents the filenumber of the file you have performed the STATISTICS command on. This number can be located using the NAMES command. When you use this option, you will need to include at least one filename on the right side of the DO command or you will get an error statement.

As an example of how these special arguments can be used, let's first use the STATISTICS command on file "TOMATO". (For the correct procedure for executing STATISTICS, see the instructions for that command in this section.) Now, let's multiply each of the data values in file "ONION" by the minimum data point value in "TOMATO" and store the results in a new file named "HI".

To do this, we must first find out the filenumber of "TOMATO" so that it can be used to identify "TOMATO" in the expression in the DO command. The command NAMES informs us that "TOMATO" is in file location number "2". We can now write the following DO command to accomplish our task:

```
COMMAND>  DO HI = ONION * MIN(2) :
```

Using the NAMES command will now show that a new file, "HI", has been created and placed in file location number "5". A LISTing of this file witll provide you with the header and data values in "HI".

LIMITATIONS:

1. While the expression on the right hand side of the DO command can be of any degree of complexity, you may only have up to twenty blanks in the expression.

2. If you specify more than one file in an expression, the files must have the same number of active cases and the same periodicity.

3. If you do use more than one file in an expression, the file in which the results are stored will have the same header as that of the last file in the expression.

4. Each filename in an expression must be surrounded by blank spaces.

5. The right hand expression in a DO command _must_ have at least one filename in it.

HELP

SYNTAX: HELP :

DESCRIPTION:

The HELP function displays the command menu of Tiny Troll at any time when the command prompt is displayed:

```
COMMAND> HELP :
```

and leaves you with a new command prompt. This command is solely an informative one and does not alter or utilize any file data.

LIST

SYNTAX: LIST filename [filename ...] :
LIST : (in DEDIT mode)

DESCRIPTION:

The LIST command displays the header and contents of one or more data files which are in Tiny Troll's active file subset. As a normal command (following a command prompt), LIST followed by the active filename(s) you are interested in seeing, produces the file header(s) and a flashing cursor. The cursor indicates that Tiny Troll is waiting for you to let it know (by striking any key but reset and control-C) that you are ready for the LISTing to proceed. Once you have hit a key, Tiny Troll will LIST the data in the files in columns under the filenames. The far left hand column LISTs the date and period with which the individual pieces of data are associated. If more than one file is LISTed, the final file LISTed defines the labelling of this column.

If more than 40 character spaces are needed to display the data in the files, Tiny Troll will automatically "wrap" the columns around so that the data isn't lost. In this case, for maximum legibility, it is recommended that an 80 or 132 column printer (either serial or parallel) be used. This feature is particularly useful for the generation of data reports.

If LIST is used in the DEDIT mode, it is typed in after an edit command prompt and no filename is needed -- but you will only get a LISTing of the file that is currently being edited. This feature is useful for checking the correctness of data you have been entering or editing. (For further information on the form of the LIST command, see the instructions for DEDIT.)

LIMITATIONS:

1. The last file in the LIST command (if more than one is being LISTed) defines the years and periods which will be associated with the data in _all_ the files being LISTed, even if the labelling is not correct for the rest of the files. Similarly, if the last file in the LIST command has fewer cases than any of the other files in the command, the LISTing will cease after the last data location in the last file LISTed has been displayed. This will happen even if some parts of other files being LISTed have not yet been displayed. To avoid this, place the file with the most cases _last_ in the LIST command.

2. If more than 40 character spaces are needed, the remaining data on the line will be "wrapped" around making the information potentially difficult to read. An 80 or 132 column printer (serial or parallel) should be used.

NAME(S)

SYNTAX:

```
NAMES :
NAME old-filename new-filename :
```

DESCRIPTION:

The command NAME(S) has two forms: one which simply displays the NAMES of the files active in Tiny Troll, along with their associated filenumbers, and one which allows you to change the NAME of a particular file.

The first form of the command is the word "NAMES" followed by a colon and return:

```
COMMAND>  NAMES :
```

If you have been following all of the examples up to this point, Tiny Troll should now respond with the following:

```
 1.  HOWDY
 2.  TOMATO
 3.  ONION
 4.  SUM
 5.  HI
 6.
 7.
 8.
 9.
10.
COMMAND>
```

These are all the files that are currently in Tiny Troll. This form of the command returns a command prompt and is strictly informational.

The second form of the command also provides a display of the active files in Tiny Troll, but also gives you the chance to alter the NAME of any of the active files. This is done by first typing the word "NAME" but don't follow it with a colon, just a return:

```
COMMAND>  NAME
```

To this Tiny Troll responds:

```
 1.  HOWDY
 2.  TOMATO
 3.  ONION
 4.  SUM
 5.  HI
 6.
 7.
 8.
 9.
10.
OLD FILENAME:
```

You should now type in the filename that you want to change.

```
OLD FILENAME:  TOMATO
```

Tiny Troll will now ask for the new filename you want to use:

```
NEW FILENAME:
```

So, give the name to Tiny Troll:

```
NEW FILENAME:  SAUCE
```

Tiny Troll will now return the command prompt to you indicating that the filename of "TOMATO" has been changed to "SAUCE" and that Tiny Troll is now ready to go on to new things. To prove this to yourself, use the first form of the command, NAMES, to display the active files in Tiny Troll's active file subset. Is "SAUCE" where "TOMATO" was? Now LIST "SAUCE" and you will see that, while the filename has changed, the values in the file are unchanged.

LIMITATIONS:

1. Only one filename can be changed at a time.

2. New filenames must conform to the original filename rules -- eight or less charactors beginning with a letter and with no intervening blanks.

3. Be careful not to use as a new filename the name of another file already in Tiny Troll's active file subset. This will cause unpredictable results when that filename is used in future commands.

4. If you use the NAME command and enter all the words of the command on one line:

```
COMMAND>  NAME TOMATO SAUCE :
```

the display of the active files which follows will reflect the old situation, not the new one -- "TOMATO" will be shown as active. This is a peculiarity of Tiny Troll; the re-NAMing has been successful, as can be seen by a subsequent utilization of the NAMES command which will contain "SAUCE".

PR#

SYNTAX: PR#n :

DESCRIPTION:

The PR# command simply allows you to turn your printer on and off. If you want to turn the printer on, all you have to do is type in PR#1 or, instead of 1, whatever slot your printer is in:

```
COMMAND>  PR#1 :
```

If you want to turn off the printer, all you need do is type:

```
COMMAND>  PR#0 :
```

Both of these commands return the command prompt.

RANGE

SYNTAX: RANGE filename start-year start-period end-year end-period :

DESCRIPTION:

RANGE allows you to temporarily reset the active

RANGE of a file, in effect selecting a subset of consecutive data values from the original file. This is particularly useful in adjusting the file size of a particular file for use in a command (e.g., PLOT, REGRESSION, DO) where other files are also utilized and where all the files must have the same number of cases. This command is temporary in the sense that the original file RANGE can be restored by simply resetting the RANGE of the file to the original start year and period and end year and period. DEDIT and SPECIFICATIONS also can be used to alter the number of cases in a file but these changes will then be permanent. (Once you have ERASEd values in a file they are gone. Of course, if you have previously SAVEd the file on diskette, you can always re-LOAD it.)

Let us assume that the file "SAUCE" (remember, this used to be called "TOMATO" but you changed the NAME) is still active in you system with a periodicity of 1, start year of 1, and start period of 1, and with data values of 15, 22, 43, 10, and 8. A LISTing of "SAUCE" will show, in the header, that the current RANGE includes all of the cases in "SAUCE". The second line shows the number of cases in the file and the third line gives the number of cases in the active subset of "SAUCE". At the moment, they are both the same: 5. If, however, you only want a subset of the middle 3 cases in "SAUCE", type the following:

```
COMMAND>  RANGE
```

followed by a return. Tiny Troll responds with:

```
FILENAME:
```

to which you type:

```
FILENAME: SAUCE
```

Tiny Troll will now give you the current start year and period and end year and period of "SAUCE":

```
CURRENT START: 1:1
CURRENT END: 5:1
START YEAR:
```

The final line requests you to give Tiny Troll the year you want to start the new subset with:

```
START YEAR: 2
```

Tiny Troll responds:

```
START PERIOD:
```

to which you respond:

```
START PERIOD: 1
```

Tiny Troll will now ask for the end year and period of your subset in the same manner:

```
END YEAR: 4
END PERIOD: 1
```

The command prompt is now returned to you, so it may be a good idea to LIST the file "SAUCE" and take a look at the header. Notice that the second line is the same as it was before you used the RANGE command but that the third line tells you that the active RANGE is now 2:1 to 4:1 and that this new RANGE has 3 cases in it. This is what you wanted, right? Now you can use "SAUCE" in any Tiny Troll command you would like to and only the middle 3 cases will be utilized. Since this is a temporary command, however, Tiny Troll is holding the "SAUCE" cases which are not being used in the active subset in reserve so that you can restore "SAUCE" to its original size whenever you would like to.

To do this, you simply have to reset the RANGE to its original parameters. The RANGE command can be given in one line, so we will utilize this option in restoring "SAUCE" to its original condition:

```
COMMAND>  RANGE SAUCE 1 1 5 1 :
```

LISTing "SAUCE" will let you see that the RANGE is now back to 1:1 to 5:1 with 5 cases.

LIMITATIONS:

1. The subsets you can create using RANGE always in-

clude all of the data between the start year and period and the end year and period you specify.

2. If you use the SPECIFICATIONS command on a file on which you have previously used the RANGE command, the RANGE will be automatically reset to incorporate all of the cases in the file, active or reserved.

SPECIFICATIONS

SYNTAX: SPECIFICATIONS fliename $\begin{matrix}\text{YES}\\\text{NO}\end{matrix}$ periodicity

start-year start-period :

DESCRIPTION:

The SPECIFICATIONS command allows you to permanently alter the header (start year and periodicity) which you used to define a file when it was first opened. This command does not change the data itself but only the dates with which the data values are associated.

We've been using "SAUCE" quite a bit lately so let's define a new file, using DEDIT, named "PIZZA" with a periodicity of 4, start year of 75 (remember, we don't need to use the first two digits in a year's numeric representation), start period of 1, and data values of 17, 25, 40, 27, 19, 22, and 30.

Now that you've done that, let's use the SPECIFICATIONS command to change the information in the header of "PIZZA" to a periodicity of 2, start year of 68, and start period of 2. First, type:

```
COMMAND> SPECIFICATIONS
```

Tiny Troll responds with:

```
FILENAME:
```

to which you type:

```
FILENAME: PIZZA
```

Tiny Troll now gives you the original file header of "PIZZA" and asks you if you want to change any of the information in the header:

```
PIZZA
PERIOD 4   START DATE  75:1     CASES=7
RANGE  75:1  TO  76:3  WITH 7 CASES
CHANGE ANY SPECS?
```

If you do want to change the information, type a "YES" and, if not, type a "NO". A "NO" will get you out of the SPECIFICATIONS command mode. Typing a "YES":

```
CHANGE ANY SPECS?
```

will elicit requests from Tiny Troll for the new information which you should supply:

```
PERIODICITY: 2
```

```
START YEAR: 68
START PERIOD: 2
```

Tiny Troll will end this sequence by displaying the new file header for you:

```
PIZZA
PERIOD 2   START DATE  68:2    CASES=7
RANGE  68:2  TO  71:2  WITH 7 CASES
COMMAND>
```

Notice that the number of cases in "PIZZA" is the same as before. A LISTing of "PIZZA" will also show that the order of the data has remained the same; all that has changed are the dates associated with the data. Instead of the first data being associated with 75:1, it is now associated with 68:2. The rest of the file has now been similarly re-allocated according to the new SPECIFICATIONS.

The above command could also have been given in one line as follows:

```
COMMAND>  SPECIFICATIONS PIZZA YES 2 68 2 :
```

You may also use the word "CHANGE" (or "CH") instead of "YES":

```
COMMAND>  SPECIFICATIONS PIZZA CHANGE 2 68 2 :
```

When you do enter the whole command in one line, Tiny Troll will respond with two file headers; first, the

the original and, second, the new one.

The SPECIFICATIONS command is particularly useful to tailor a file so that it can be used in certain commands (e.g., PLOT, REGRESSION, DO) where it is essential that the files used have the same periodicity and number of cases, and where it is helpful (for labelling purposes) that they have the same start date.

LIMITATIONS:

1. Once the SPECIFICATIONS of a file are changed, the old header information is deleted from the system and cannot be recovered unless they are re-entered into Tiny Troll.

CATALOG

SYNTAX: CATALOG :

DESCRIPTION:

The CATALOG command displays the contents of any diskette which is in the disk drive which is currently in the default SLOT number and DRIVE number, without losing whatever files you already happen to have active in Tiny Troll. This command is designed to help you find data files which have been previously SAVEd on the diskette; you may not be sure what diskette the file you desire is on or you may have forgotten the correct spelling of the filename. This command can be used after any command prompt:

```
COMMAND>  CATALOG :
```

LOAD

SYNTAX: LOAD filename :

DESCRIPTION:

The LOAD command allows you to LOAD any data file which has been SAVEd on diskette from a previous Tiny Troll session. The format for doing this is exactly the same as is described in your DOS 3.2 manual for LOADing any program from diskette into the Apple II except that you can only use those files which are preceded by a "T" (for "TEXT") in the disk catalog and that you must conclude the command with a colon:

```
COMMAND>  LOAD FILEA :
```

Unlike many of the other Tiny Troll commands, LOAD is not interactive and must, therefore, be entered in one complete command line and only <u>one</u> file can be LOADed at a time.

Note: Before LOADing a file into Tiny Troll, it is highly suggested that you use the NAMES command to make certain that you have room for a new file and that you will not be deleting a file that you do not want to lose.

SAVE

SYNTAX: SAVE filename :

DESCRIPTION:

The SAVE command in Tiny Troll serves the function of storing data files onto diskette for later usage. The format is the same as that used for SAVing any file in DOS 3.2 except that a concluding colon is needed. As in the case of the LOAD command, the SAVE command must be entered all on one line:

```
COMMAND>  SAVE TOMATO :
```

In order to distinguish a data file from other files on a diskette's catalog, a "T" will precede data files in the catalog display.

Note: It is highly recommended that you SAVE any newly created files onto a diskette as soon as possible in order to safeguard the file from being accidentally lost.

SLOT, DRIVE

SYNTAX: SLOT n :

DRIVE n :

DESCRIPTION:

These commands allow a user with two or more disk drives to select the SLOT and DRIVE to be used in SAVing or LOADing a file from diskette or for viewing a diskette's CATALOG.

Tiny Troll will automatically come up in SLOT 6, DRIVE 1. If you desire to change either of these locations, all you need do is place the number of the SLOT or DRIVE you want after the appropriate command. For example, if you want to use a disk drive in SLOT 7, DRIVE 2, just enter:

```
COMMAND> SLOT 7 : DRIVE 2 :
```

and follow it with whatever command you want. This SLOT and DRIVE location will now become the default file for the rest of the Tiny Troll session unless you change it.

Further information on the utilization of these commands can be found in your DOS 3.2 manual.

STATISTICS

SYNTAX: STATISTICS filename :

DESCRIPTION:

The STATISTICS command calculates the mean, variance, and standard deviation of the data in an active data file along with indicating both the maximum and minimum values of the file. For example, let's use the STATISTICS command on the "PIZZA" file we defined earlier. (If you don't happen to have it handy, it has a periodicity of 2, start year of 68, start period of 2, and data values of 17, 25, 40, 27, 19, 22, and 30.) You can enter the whole command on one line (as is the case with all Tiny Troll commands) or you can use STATISTICS interactively:

```
COMMAND>  STATISTICS
FILENAME: PIZZA
```

You will notice that Tiny Troll first displays the file header and then the following:

```
MAX = 40          MIN = 19
MEAN = 26.6
VARIANCE = 16.16
STD DEVIATION = 4.02
```

The STATISTICS command also produces two other pieces of output which are not displayed but which are available for use in the right hand expression of the DO command. These are the sum of the data points in the file and the sum of squares of the file. In addition, MIN and MAX are also available to the DO command. (For more on the usage of these results, see the instructions for the DO command.)

LIMITATIONS:

1. Only one file can be used with the STATISTICS command at a time.

2. The results of the STATISTICS command cannot be held in Tiny Troll for later use but must be copied down and re-entered into the system each time they are needed. The only exceptions to this rule are the output (MIN, MAX, SUM, SS) which are addressable in the DO command.

PLOT

SYNTAX: PLOT [ZERO] filename [filename ...] :

DESCRIPTION:

The PLOT command is a powerful visual tool for seeing, graphically, the fluctuations within one file or the relationship between two or more files. It provides you with the ability to PLOT, in high resolution graphics, the contents of up to six data files of equal periodicity and RANGE. This command provides automatic scaling of data, automatic axis labelling, and a PLOT legend.

The X axis represents the dates associated with the PLOTted files, and the Y axis represents the values of the data points in the files. If more than one file is PLOTted, each file is assigned its own symbol to help identify the graph lines and the identifying symbol is placed in the PLOT legend below the graph. If you want to make sure that the graph has a zero line in it, all you have to do is place the word "ZERO" after the word PLOT and before the first filename in the command.

PLOT is not is an interactive command so you must give Tiny Troll the whole command in one line. Also, if an attempt is made to PLOT two or more files of unequal periodicities and/or RANGEs, an error message will be generated (see Appendix B). To avoid this situation it is necessary for you to adjust the periodicities and the RANGEs of your files using the SPECIFICATIONS and RANGE commands. Remember, the RANGE command alters your files only temporarily while the SPECIFICATIONS command permanently alters them. If you do use the SPECIFICATIONS command and you don't want to lose your original file header information, it is recommended that you SAVE the file to be altered prior to using SPECIFICATIONS. The following example should be helpful.

We will use two files which you should still have in Tiny Troll's active file subset, "SAUCE" and "PIZZA". If you attempt to PLOT "SAUCE" and "PIZZA" in the same command, as they are:

```
COMMAND>  PLOT SAUCE PIZZA :
```

Tiny Troll will respond with an error statement which indicates that either the periodicities or the RANGEs of the files are not the same. To check this out, LIST both of the files and examine the headers:

```
SAUCE
PERIOD 1   START DATE 1:1    CASES=5
RANGE  1:1  TO 5:1  WITH 5 CASES


PIZZA
PERIOD 2   START DATE 68:2   CASES=7
RANGE  68:2 to 71:2   WITH 7 CASES
```

The above LISTing shows you that the headers of the two files are entirely different. When you encounter a situation such as this, it will be necessary for you to utilize the RANGE and/or the SPECIFICATIONS commands in order to be able to PLOT the files together. You will have to end up with the same file headers for all of the files you are PLOTting. In this case, the process will entail changing all the information in at least one of the files.

The first thing we have to do is select the periodicity and start date which we wart the files to have. Let's adjust "SAUCE" so that its periodicity and start date are the same as those of "PIZZA". To do this it will be necessary to use the SPECIFICATIONS command, thus altering "SAUCE" permanently:

```
COMMAND>  SPECIFICATIONS SAUCE CH 2 68 2 :
```

Now the files have the same basic header information but the RANGEs are still different with "PIZZA" having a

RANGE of 68:2 to 71:2 and "SAUCE" having a RANGE of 68:2 to 70:2. "PIZZA" has two more cases in it than does "SAUCE". This can be remedied by using the RANGE command on the larger file ("PIZZA") to create a subset with the same RANGE as the smaller file ("SAUCE"):

```
COMMAND>  RANGE PIZZA 68 2 70 2 :
```

If you now LIST "PIZZA" and "SAUCE" you will find that their RANGEs and periodicities are the same:

```
PIZZA
PERIOD 2   START DATE 68:2   CASES=7
RANGE  68:2 TO 70:2  WITH 5 CASES

SAUCE
PERIOD 2   START DATE 68:2   CASES=5
RANGE  68:2  TO 70:2  WITH 5 CASES
```

You can now PLOT "PIZZA" and "SAUCE" (in any order):

```
COMMAND>  PLOT PIZZA SAUCE :
```

Tiny Troll will now graph the two files for you with "PIZZA" being PLOTted first and "SAUCE" second; the legend below the graph will tell you which line is which. If your data had had values of 100 or above, or of less than 1.0, the Y axis would have been automatically adjusted according to the exponential value of the data so that there would be four or less digits on the axis labels. The exponent by which the Y axis has been adjusted would then be placed in the upper left hand corner of the CRT screen.

If hard copy of a PLOT is desired, a special graphics type printer (serial or parallel), such as the Integral Data or the Axiom, must be used, and you will have to arrange for software to dump the high resolution grahics screen to the printer.

LIMITATIONS:

1. A maximum six files can be PLOTted on the same graph.

2. All of the files utilized in PLOT must have the same RANGE and periodicity.

REGRESSION

SYNTAX: `REGRESSION filename filename [filename ...] :`

DESCRIPTION:

The REGRESSION command is perhaps the most valuable Tiny Troll function for the business or financial analyst. It provides you with the ability to discover how predictive one or more independent variables are of the values of a dependent variable. This is particularly helpful in the development of models and for forecasting. Tiny Troll's REGRESSION command utilizes the ordinary least squares estimation of the coefficients of the variables.

In addition, REGRESSION provides you with a measure of the goodness of fit of the REGRESSION, a measure of the colinearity of the REGRESSION, a matrix of the variance-covariance interaction, the standard error of the REGRESSION, a matrix of the correlation between the variables, measures of the contributory significance of the variables, and the fitted values and residuals.

If you are not very familiar with the ins and outs of multiple linear regression, it is suggested that you read carefully Chapter 3, "Why Regress?", before using this command. For those of you who are familiar with the subject, and are ready to plow ahead with the command, the following will lead you through a sample REGRESSION.

You can have as many as nine independent variables with one dependent variable; all of these variables must have the same periodicity and RANGE. In this example, we will use three variables (data files) with periodicity of 4, and start date of 75:1. Using this information, create the files "GNP" (for Gross National Product), "CON" (for Consumption), and "R" (for Prime Interest Rate).

The following table contains the data values you should enter for the variables:

<u>DATE</u>	<u>GNP</u>	<u>CON</u>	<u>R</u>
75:1	601.6	387.9	3.93
75:2	610.4	393.4	3.8
75:3	622.5	400.3	3.92
75:4	636.6	409.2	4.38
76:1	649.1	415.3	4.59
76:2	655.0	415.1	4.5
76:3	660.2	421.3	5.37
76:4	668.1	420.7	4.96
77:1	666.6	424.2	4.26
77:2	671.6	430.3	3.54
77:3	678.9	431.6	4.42
77:4	683.6	434.3	4.97

After creating these files, you should LIST them to make certain there are no mistakes.

In this example, we will attempt to discover how predictive "CON" and "R" are of "GNP". We will not try to interpret the results of the REGRESSION command but will leave that to you (with the help of Chapter 3). The format for entering this request into Tiny Troll is quite simple:

```
COMMAND>  REGRESSION GNP CON R :
```

The first variable entered into a REGRESSION command is <u>always</u> the dependent variable and the independent variables can follow in any order. You should also note that REGRESSION is not interactive and that the command must, therefore, be entered all on one line.

Tiny Troll will respond to the above command with:

```
                    ***MULTIPLE REGRESSION***
PRELIMINARY COMPUTATIONS
                              COMPUTING
PERIODICITY 4
RANGE  75:1  TO 77:4  WITH 12 CASES
```

You will notice that the word "COMPUTING" is flashing which indicates that Tiny Troll is thinking. This could go on for a few minutes depending upon how many variables and cases you have entered; there are an extremely large number of individual computations in a REGRESSION procedure which can take even your Apple II a while to perform. When Tiny Troll is finished with the computations, it will display:

```
                    :::: REGRESSION ANALYSIS ::::


DEPENDENT VARIABLE IS #1 GNP
CORRELATION COEFFICIENTS
VAR.   2     3     1
2      1     .44   .99
3      .44   1     .5
1      .99   .5    1
WANT VARIANCE-COVARIANCE MATRIX?
```

This gives you the correlation matrix of the variables. Notice that the variables are not identified by name but, rather, by the number of the file to which each was assigned when first entered into the system. The filenumbers range from one to ten with each successive file being assigned to the next available filenumber. Assuming that "GNP" was entered first, and that Tiny Troll was booted just prior to entering "GNP", it is placed in filenumber 1. Similarly, "CON" and "R" are in files 2 and 3, respectively. You can obtain a display of which files are associated with which filenumbers by using the NAMES command.

After you have viewed the correlation matrix, Tiny Troll asks you whether you want to see the matrix of the variance-covariance interaction of the variable coefficients which Tiny Troll has computed. (If you need a record of all these results, it is suggested that you use a printer.) If you don't want to see this matrix, type "N" and Tiny Troll will next display the table of results from REGRESSION and its attendant statistics, otherwise it will print the following:

```
VARIANCE-COVARIANCE MATRIX
FOR REGRESSION COEFFICIENTS
CFT  0      2     3
0  785.14 -1.99 9.2
2   -1.99  .01  -.07
3    9.2  -.07  4.28
```

```
CONTINUE?
```

"CONTINUE?" is really just a pause to allow you to study the above matrix. When you want to see more just hit return and the results table will be displayed:

```
VARIABLE          COEFF.        STD.ERR.          T-STAT
==========================================================

0  CONSTANT:   -80.42          28.02            -2.87

2  CON:           1.72           .07              23.22
3  R:
                  3.7            2.07              1.79
==========================================================

R-SQ = .99                CORRECTED R-SQ = .98

F(2,9) = 358.54                NOB = 12

SER = 3.32                     SSR = 99.27

DURBIN-WATSON STAT. = 2.8
```

```
WANT FITTED VALUES AND RESIDUALS?
```

Once again, Chapter 3, "Why Regress?", will help you to understand the statistics given above. It might be helpful, though, to know what some of the abbreviations stand for:

NOB let's you know the number of cases that are being analyzed for each variable. This will, of course, be the same for all of the variables.

SER stands for the standard error of the REGRESSION.

SSR stands for the sum of squares of the REGRESSION.

Next Tiny Troll wants to know if you would like to see the fitted values and residuals. If not, type "N" and the command prompt will be returned to you indicating that you are now out of REGRESSION mode; if you type "Y", Tiny Troll will display the following:

```
                          VAR#1 GNP

#         ACTUAL          FITTED          RESIDUAL

1         601.6           601.52          .08

2         610.4           610.5           -.1

3         622.5           622.82          -.32

4         636.6           639.83          -3.23

5         649.1           651.1           -2.0

6         655.0           650.43          4.57

7         660.2           664.31          -4.11

8         668.1           661.76          6.34

9         666.6           665.19          1.41

10        671.6           673.02          -1.42

11        678.9           678.52          .38

12        683.6           685.2           -1.6

SAVE THE RESIDUALS?
```

If you want to SAVE the residuals for future use (such as PLOTting) then you must SAVE the residuals at this point since you cannot access them directly from the REGRESSION results. If you do SAVE the residuals, Tiny Troll will automatically SAVE them on to diskette and respond with:

```
FILENAME:
```

to which you enter the filename that you would like to use.

which tells you the name of the text file under which the residuals are SAVEd and that the REGRESSION function is now completed.

LIMITATIONS:

1. You can have up to nine independent variables REGRESSed on to one dependent variable.

2. You cannot SAVE the matrices or tabled statistics for future use. If you want to use them in later computations, you should obtain hard copy and then re-enter the data into the system when you need to.

3. The files must all have the same periodicity and RANGE.

4. The entire command must be entered on one line.

MULTIPLE COMMAND LINES

As has been noted repeatedly throughout this manual, commands can be entered into Tiny Troll in a variety of lengths and complexities. You can enter part of a command, an entire command, or a series of commands in one command line. When you become more familiar with Tiny Troll's command structure you will undoubtedly want to take advantage of the option to enter more than one command on a command line.

You can string as many commands on a single line as you want, subject to a few limitations:

1. Each complete command must be separated by a

colon with spaces both before and after the colon.

2. You can have a maximum of twenty-five words in your command line, where a word is any combination of letters, numbers, or symbols surrounded by spaces. (A colon is considered a word.)

CHAPTER III

WHY REGRESS?

The REGRESSION command of Tiny Troll affords the business and financial user, in particular, access to a very powerful tool for developing models for understanding past trends and for forecasting future trends. This chapter provides a basic introduction to multiple linear regression and helps you to understand and interpret the output of the REGRESSION command.

We are grateful to the National Association of Business Economists, and especially to Mr. Donald L. McLagan, vice president of Data Resources, Inc., for permission to reprint the following article by Mr. McLagan, "A Non-econometrician's Guide to Econometrics", which appeared in the May, 1973 edition of Business Economics. This article gives a cogent explanation of the basics of multiple linear regression describing most of the statistics which are output by the REGRESSION command. Immediately following the article, the few items in the Tiny Troll output which are not mentioned in the article will be explained.

If you are interested in a more rigorous explanation of multiple linear regression, please refer to the books named in the footnote on the last page of Mr. McLagan's article.

A Non-econometrician's Guide to Econometrics

Donald L. McLagan

Vice President
Data Resources, Inc.

A quick reference on the concepts, techniques, and pitfalls for selecting regression equations. The author avoids being rigorously precise and thereby reaches behind the technical jargon in providing useful guidelines for applying regression analysis to business and economic forecasting.

ECONOMETRICS is being used increasingly by business economists to anticipate the effects of economic change on their business. Techniques which were confined within the walls of academic institutions ten years ago are now the analytic allies of corporate, financial and government forecasters. Unfortunately, the transition of econometrics from academic to business applications has not been accompanied by supporting documentation of terminology, expected benefits and shortcomings oriented towards the practical user.

The purpose of this article is to provide a quick reference to the concepts, pitfalls and techniques for selecting regression equations – regression analysis being the principal tool of econometrics. It is not the intent of this article to be rigorously precise. Rather, it attempts to reach behind the technical jargon to find useful guidelines for the application of regression to business and economic forecasting.[1]

SOME REGRESSION CONCEPTS AND TERMINOLOGY

LEAST SQUARES (Multiple Regression)

LEAST SQUARES is a method of developing an equation which relates one variable (such as a company's sales) to one or more other variables which should explain the first (such as price, economic demands, competition, etc.). This method is mathematically contrived so that the resulting combination of explanatory variables produces the smallest error between the historic actual values and those estimated by the regression.

The output of a *LEAST SQUARES* analysis provides an equation for the variable being explained (the left-hand or dependent variable). In this equation, the left hand variable equals the sum of the explanatory (the right hand or independent) variables, each multiplied by a regression *ESTIMATED COEFFICIENT*.

Example 1 is in the format of a timesharing computer program print-out which estimates a Retail Company's Sales *(SALES)* as a function of Disposable Income *(YD)* and the Unemployment Rate *(RU)* with *LEAST SQUARES*. The equation indicates that *SALES* equals a constant of −2.1 plus an *ESTIMATED COEFFICIENT* of 14.8 times Disposable Income minus 173.0 times an *ESTIMATED COEFFICIENT* of the Unemployment Rate. *LEAST SQUARES* analysis produces several statistics which indicate the success achieved with the estimating equation. These statistics are shown in the print-out and are discussed below.

R-Bar Squared

The *R-BAR SQUARED* is the best known indicator of the success of a *LEAST SQUARES* fit. Basically, the *R-BAR SQUARED* measures the percentage of the change in the left hand (or dependent) variable in the equation which has been ex-

---

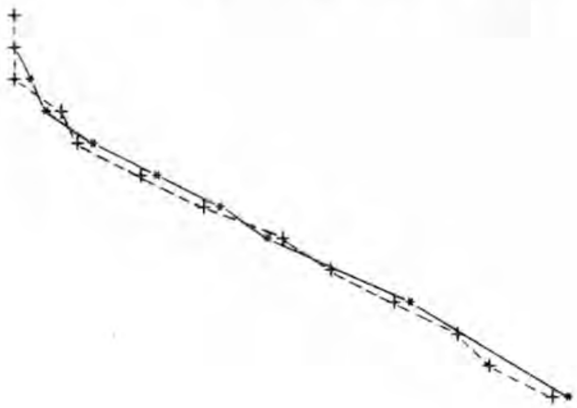
[1] See end of text for footnotes.

EXAMPLE 1

Resulting Regression Equation

$SALES = -2.1 + 14.8\,YD - 173.0\,RU$

```
ORDINARY LEAST SQUARES

  FREQUENCY ANNUAL
  INTERVAL   59: 1 TO 71: 1
  LEFT-HAND VARIABLE: SALES

RIGHT-HAND        ESTIMATED        STANDARD           T-
 VARIABLE         COEFFICIENT      ERROR            STATISTIC

  CONSTANT          -2.10487        258.938         -.812885E-02

  YD                14.8191         0.263968         56.1398

  RU               -173.022          35.0928         -4.93043

R-BAR SQUARED:    0.9971
DURBIN-WATSON STAT. (ADJ. FOR 0. GAPS):    1.9724
STANDARD ERROR OF THE REGRESSION:          112.367

DATE        ACTUAL        FITTED                  PLOT: * = ACTUAL;   + = FITTED

59          4036.         4054.
60          4134.         4226.
61          4268.         4241.
62          4603.         4744.
63          5116.         5018.
64          5740.         5598.
65          6390.         6231.
66          6805.         6927.
67          7330.         7428.
68          8198.         8140.
69          8863.         8793.
70          9262.         9355.
71          10006         9996.
```



plained by changes in the explanatory variables. In the case of a perfect explanation, the *R-BAR SQUARED* will be 1.0. Generally, the higher the *R-BAR SQUARED*, the better the equation.

In Example 1, 99.71% of the change in *SALES* (over the interval 1959 to 1971) has been explained by a constant and changes in Disposable Income and changes in the Unemployment Rate. There are other indicators of the success of the *LEAST SQUARES* fit which must be looked at beyond the *R-BAR SQUARE*. Among these are the *T-STATISTICS* of the coefficients, the *STANDARD ERROR OF THE REGRESSION*, and the *DURBIN-WATSON STATISTIC*.

Fitted and Actual Values

FITTED values are estimates of the *LEFT HAND VARIABLE* developed by using the historic explanatory variables in the regression equation. Thus, the *FITTED* values are the predictions which would have been achieved had the regression equation been used in the past.

In Example 1, the *FITTED* value for 1959 is 4054 versus the *ACTUAL* value of 4036. Comparing the *FITTED* values with the *ACTUAL* values is often called "ex post" analysis (i.e. with the benefit of hindsight since the *FITTED* values are from the interval used to develop the regression equation in the first place).

Standard Error of the Regression

The *STANDARD ERROR OF THE REGRESSION* gives a measure of how close the *FITTED* values have been to the *ACTUAL* values in the past. When regression analysis is being used to develop equations, it is desirable to have the *STANDARD ERROR OF THE REGRESSION* as small as possible. Also, this statistic may be used to gain some idea of the forecasting accuracy which can be expected.

In Example 1, the *STANDARD ERROR OF THE REGRESSION* is 112. This means that:

- in 67% of the historical observations, the estimate is within ± 1 *STANDARD ERROR* of the *ACTUAL* (± 112 in Example 1).
- in 95% of the historical observations, the estimate is within ± 2 *STANDARD ERRORS* of the *ACTUAL* (± 224 in Example 1)
- in 99% of the historical observations, the estimate is within ± 3 *STANDARD ERRORS* of the *ACTUAL* (± 336 in Example 1).

T-Statistic

The *T-STATISTIC* shows the significance of each explanatory variable in predicting the dependent variable. It is desirable to have as large (either positive or negative) a *T-STATISTIC* as possible for each explanatory variable. Generally, any statistic greater than +2.0 or less than −2.0 is acceptable. If the *T-STATISTIC* is between −2.0 and +2.0 for any independent variable, that variable is not contributing significantly to explaining the dependent variable. Explanatory variables with low *T-STATISTICS* can usually be eliminated from the regression without substantially decreasing the *R-BAR SQUARED* or increasing the *STANDARD ERROR OF THE REGRESSION*. (Basically the *T-STATISTIC* is a measure of how unlikely it is that there is *no* relationship between the dependent variable and an independent variable. Thus a high *T-STATISTIC* would make it very unlikely that the indicated relationship is just a fluke).

In Example 1, the *T-STATISTICS* on Disposable Income (56.1) and the Unemployment Rate (−4.9) are both well beyond the threshold. This means that both variables contribute significantly to explaining the Retail Company's Sales. The constant (−.008) is not significant and could be dropped from the equation.

Standard Error (of the coefficients)

The coefficients of the regression are estimated statistically. The *ESTIMATED COEFFICIENTS* are the best estimates of the coefficients for each explanatory variable. The *STANDARD ERROR* of the coefficient gives an estimate of the range in which that coefficient may "actually" be. Like the *T-STATISTIC* the *STANDARD ERROR* of the coefficient indicates how significant each independent variable is in explaining the dependent variable. In fact, the *T-STATISTIC* is developed by dividing the *ESTIMATED COEFFICIENT* for a variable by its *STANDARD ERROR*. Thus, the *T-STATISTIC* really measures how many *STANDARD ERRORS* the coefficient is away from 0. If the *ESTIMATED COEFFICIENT* of an independent variable were 0, no relationship between the dependent variable and this particular independent variable would be indicated.

In Example 1, the *STANDARD ERROR* of the constant (258.9) is considerably greater than the *ESTIMATED COEFFICIENT* of the constant (−2.1). This means that the constant is not significantly different from 0. The *STANDARD ERROR* for Disposable Income (0.26) and for the Unemployment Rate (35.1) are small relative to their *ESTIMATED COEFFICIENTS* (14.8 and 173.0 respectively) indicating that both explanatory variables have a substantially different from zero effect in explaining *SALES*, and that the association would not be due to chance.

Durbin-Watson Statistic

See the discussion on the pitfall of *AUTOCORRELATION*.

Correlation

Any two variables may vary with each other. The extent to which they do so is called their *CORRELATION*. If the positive change in one variable is directly and perfectly related to the positive change in another variable, then the *CORRELATION* between these two variables will be +1.0; if one decreases as the other increases, then the *CORRELATION* will be −1.0. If the relationship is imperfect and positive, the correlation will be between 0 and 1.0. If there is no relationship between the change in one variable and the change in another, then the *CORRELATION* will be 0.0.

CORRELATION between a set of variables is often displayed in a *CORRELATION MATRIX* (see Example 2) which is read in much the same way as distances between cities on a roadmap. By locating the second number in the third row, you have found the correlation between the second and third variable.

EXAMPLE 2

```
CORRELATION MATRIX

           SALES     YD        C         RU

SALES      1.0000

YD         0.9958    1.0000

C          0.9970    0.9997    1.0000

RU        -0.4807   -0.4124   -0.4239    1.0000
```

This example shows the *CORRELATION* between the Retail Company's Sales *(SALES)*, Personal Consumption Expenditures *(C)*, Disposable Income *(YD)* and the Unemployment Rate *(RU)*. Note that although the Unemployment Rate has a relatively low *CORRELATION* with *SALES* (−0.48), the regression results of Example 1 show that Unemployment does contribute significantly to Explaining *SALES* when combined with Disposable Income. This is one reason why regression equations are more useful than simple *CORRELATIONS* for forecasting.

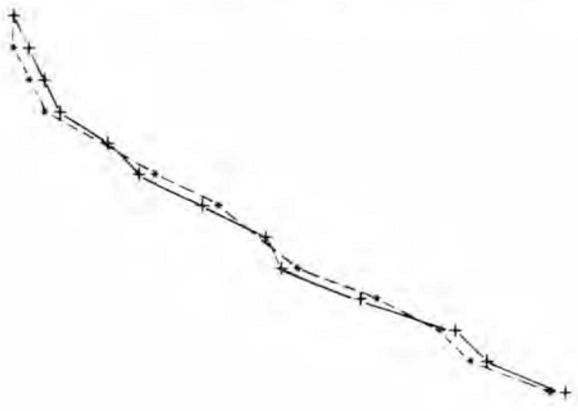
REGRESSION PITFALLS: RECOGNIZING AND DEALING WITH THEM

The two major pitfalls encountered in regression analysis are *COLLINEARITY* and *AUTOCORRE-*

EXAMPLE 3

Note the unexpected *negative* estimated coefficient on Disposable Income (YD).

```
ORDINARY LEAST SQUARES

  FREQUENCY ANNUAL
  INTERVAL    59: 1 TO 71: 1
  LEFT-HAND VARIABLE:  SALES

RIGHT-HAND            ESTIMATED            STANDARD             T-
 VARIABLE            COEFFICIENT             ERROR           STATISTIC

  CONSTANT            -1937.13              355.049           -5.45595

  C                     40.4550              15.3383            2.63752

  YD                   -19.8445              13.3504           -1.48643

R-BAR SQUARED:    0.9941
DURBIN-WATSON STAT.  (ADJ. FOR 0.  GAPS):   0.9343
STANDARD ERROR OF THE REGRESSION:           159.836

DATE          ACTUAL          FITTED               PLOT:  * = ACTUAL;   + = FITTED

59            4036.           3959.
60            4134.           4274.
61            4268.           4390.
62            4603.           4781.
63            5116.           5204.
64            5740.           5600.
65            6390.           6182.
66            6805.           6771.
67            7330.           7128.
68            8198.           8026.
69            8863.           8916.
70            9262.           9331.
71            10006           10190
```



LATION. The following explains what each of these pitfalls is, how to recognize it, and how to deal with it.

Collinearity

When using more than one independent variable in a regression equation, there is sometimes a high correlation between the independent variables themselves. *COLLINEARITY* is when these explanatory variables interfere with each other. It is a pitfall because the use of equations with *COLLINEARITY* may produce spurious forecasts.

COLLINEARITY can be recognized when the regression *T-STATISTICS* of two seemingly important explanatory variables are low. Often *COLLINEARITY* even causes the *ESTIMATED COEFFICIENTS* on explanatory variables to have the opposite sign from that which would logically be expected. Example 3 shows a case of *COLLINEARITY* between Disposable Income and Consumption in explaining the sales of the Retail Company. It is not reasonable to expect Sales to decrease as Disposable Income increases as indicated by the negative coefficient on Disposable Income.

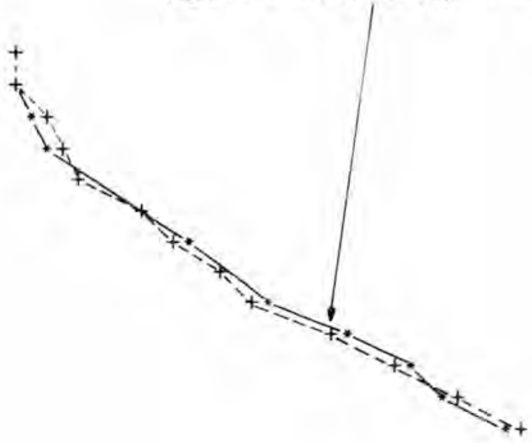
The *COLLINEARITY* becomes apparent when the variables are compared in a correlation matrix. Looking back at the *CORRELATION MATRIX* of Example 2, shows that Disposable Income and Consumption are, in fact, highly correlated (0.9997).

There are two ways to get around the problem of *COLLINEARITY*. First, one of the highly correlated variables may be dropped from the regression. In Example 4, consumption was dropped from the regression. Note that the *R-SQUARED* and *STANDARD ERROR OF THE REGRESSION* are not affected appreciably, but the *T-STATISTIC* on Disposable Income improves dramatically now that consumption no longer interferes with it.

EXAMPLE 4

Note the low DURBIN-WATSON statistic indicating positive AUTOCORRELATION of the errors.

```
ORDINARY LEAST SQUARES

  FREQUENCY ANNUAL
  INTERVAL   59: 1 TO 71: 1
  LEFT-HAND VARIABLE: SALES

RIGHT-HAND          ESTIMATED          STANDARD          T-
 VARIABLE          COEFFICIENT           ERROR        STATISTIC

  CONSTANT           -1123.73            218.429       -5.14459

  YD                  15.3558            0.424692       36.1576

R-BAR SQUARED: 0.9909
DURBIN-WATSON STAT. (ADJ. FOR 0. GAPS):   0.8383
STANDARD ERROR OF THE REGRESSION:          198.448

DATE        ACTUAL         FITTED                   PLOT: * = ACTUAL;  + = FITTED

59          4036.          4056.
60          4134.          4251.
61          4268.          4472.
62          4603.          4792.
63          5116.          5089.
64          5740.          5604.
65          6390.          6143.
66          6805.          6736.
67          7330.          7266.
68          8198.          7952.
69          8863.          8618.
70          9262.          9464.
71          10006          10307
```



The second way around *COLLINEARITY* is to change the structure of the equation. One approach is to divide both left and right hand variables by some series which will leave the basic economic logic but remove the collinearity. For the Retail Company, population might be a good candidate. Estimating Sales per capita by consumption per capita and Income per capita, though different from the original formulation (the simple level of sales), may diminish the *COLLINEARITY* problem. Another structure which usually reduces the *COLLINEARITY* is to estimate the equation on a first difference basis; that is, estimate the change in Retail Company Sales on the change in Income and the change in Consumption. Another way out is to combine the collinear variables into a new variable which is their weighted sum. Often the weights can be derived from economic reasoning.

Autocorrelation

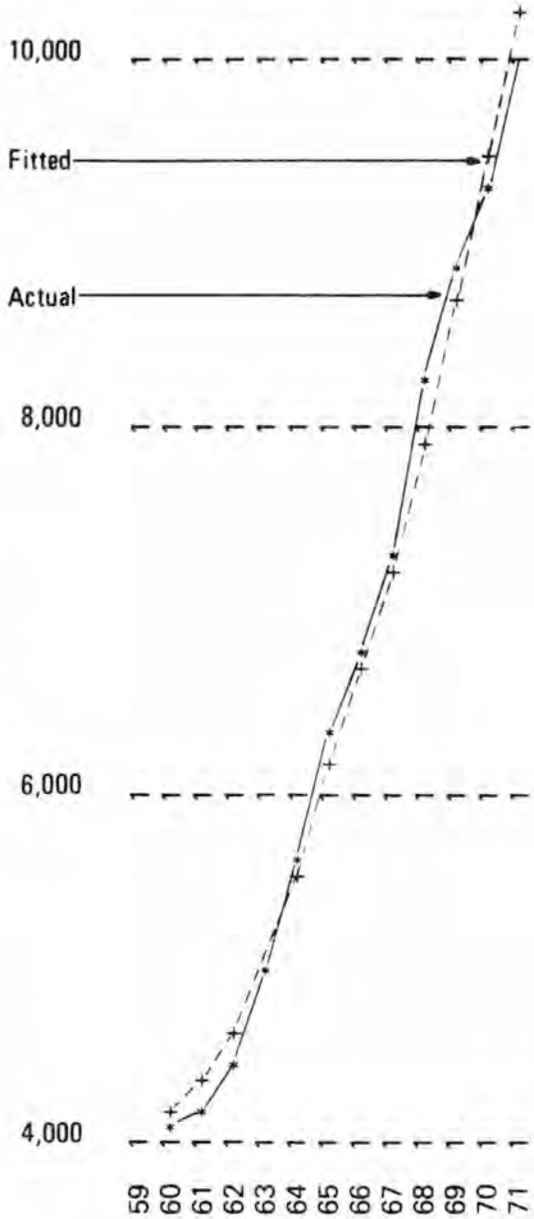
AUTOCORRELATION is the other major pitfall often encountered in regression analysis. One principal assumption of regression analysis is that the errors (between the *FITTED* and *ACTUAL* values) are independent from one observation to the next. That is, knowledge of the error in one year will not help you anticipate the error in the next year. *AUTOCORRELATION* is the case where there is a correlation between successive errors. It is a pitfall because the error between the *FITTED* and *ACTUAL* value in the last observation is apt to persist into the forecast interval.

The *DURBIN-WATSON STATISTIC* provides the standard test for *AUTOCORRELATION*. Generally, if the *DURBIN-WATSON STATISTIC* is between 1.5 and 2.5, there is not serious *AUTOCORRELATION* in a regression equation. If the *DURBIN-WATSON* is below 1.5, there may be positive *AUTOCORRELATION* between errors. In this case, a positive error in one observation will usually indicate a positive error in the subsequent observation. If the *DURBIN-WATSON* is above 2.5, negative *AUTOCORRELATION* may be indicated, and the errors generally alternate negative and positive.

Example 4 illustrates serious positive *AUTOCORRELATION*. Note that the *DURBIN-WATSON STATISTIC* is 0.8383 in this regression. The *FITTED* and *ACTUAL* values shown in expanded scale in Example 5 illustrate the persistence of the errors. If this equation was used to forecast 1972 Sales, the resulting estimate would probably be high because of the positive *AUTOCORRELATION* and the overestimate in 1971.

EXAMPLE 5

DATE	ACTUAL	FITTED	ERROR
59	4036.00	4056.03	20.0288
60	4134.00	4251.49	117.493
61	4268.00	4472.31	204.310
62	4603.00	4792.37	189.372
63	5116.00	5089.31	−26.6915
64	5740.00	5603.61	−136.393
65	6390.00	6143.27	−246.728
66	6805.00	6736.19	−68.8075
67	7330.00	7265.80	−64.1996
68	8198.00	7951.53	−246.469
69	8863.00	8617.81	−245.194
70	9262.00	9464.50	202.497
71	10006.0	10306.8	300.780



Usually *AUTOCORRELATION* indicates that there is an important part of the variation of the dependent variable which has not been explained. The best solution to this problem is to search for other explanatory variables to include in the regression equation. In the case of the Retail Company's Sales, including the Unemployment Rate with Disposable Income reduces the *AUTOCORRELATION* to an insignificant level (the *DURBIN-WATSON* in Example 1 is 1.9724).

Sometimes no additional variables can be identified to reduce *AUTOCORRELATION* to an insignificant level. In this case, forecasts from the equation with *AUTOCORRELATION* should be modified to reflect the error in the last observation. One technique for phasing out the effect of the last observed error is to calculate a factor (usually called rho) which is half of the difference between the *DURBIN-WATSON* and 2.0. In Example 4, rho = (2.0−0.8383)/2 = 0.581. This factor is multiplied times the last observed error and the result is added to the forecast from the equation for the first period. In Example 4, 0.581(10010−10310) = −174, which should be added to any forecast for 1972 developed from the regression equations from Example 4. Use of the same factor (e.g. .581) times the amount added to the first forecast period (e.g. −174) provides the amount to be added to the equation result for the second forecast period (e.g. .581(−174) = −101). This process is continued through the forecast interval until the modification to the equation results is reduced to an insignificant level.

SELECTING THE BEST EQUATION

With the advent of timesharing, large data bases and English-language regression software, the problem of developing regression equations has shifted from getting *an* equation estimated to selecting among several equations which have been estimated easily. Choosing among alternative forecasting equations is somewhat like handicapping a horserace. First, the obvious losers must be eliminated, then the winner must be selected from among the remaining contenders.

Some equations fail to meet standard acceptance criteria and can be eliminated easily. Selecting the best equation is more difficult, and involves a close analysis of each equation's track record. The following principles have proven useful in selecting appropriate forecasting equations.

Eliminating the Losers

1. Does the equation make sense?

   There must be some plausible causality between the dependent and each of the independent variables. This criterion immediately eliminates sunspot-and-stock-market equations. It also eliminates equations as in Example 3 where *COLLINEARITY* causes one explanatory variable (e.g. Disposable Income) to take on the opposite sign from that which would be reasonably expected. Other tests for sensibility involve comparability of data on both the left and right hand side of the equation. If the dependent variable is in units or constant dollars, then the independent variables should not be in current dollars. If the dependent variable is seasonally adjusted, the independent variables should be also. Note that in example 1, the unemployment rate is obviously in different units than sales of the Retail Company. The equation would make more sense if the dollars of income foregone due to unemployment were used instead of the rate of unemployment.

2. Low *T-STATISTICS*

   Equations which have explanatory variables with low *T-STATISTICS* should be re-estimated or dropped in favor of equations in which all independent variables are significant. This test will eliminate equations where *COLLINEARITY* remains a problem (see the Pitfall section on *COLLINEARITY* for suggestions of alternative equation structures which may sidestep the *COLLINEARITY* problem).

3. Low *R-BAR SQUARED*

   The *R-BAR SQUARED* can be used to rank the remaining equations and to select the best three or four candidates. (Note: this approach usually produces identical selections as choosing among the remaining equations by minimizing the *STANDARD ERROR OF THE REGRESSION*.)

Choosing the Best Equation

1. Turning points and recent accuracy

   Examining the (ex post) performance of *FITTED* and *ACTUAL* values provides the easiest basis for selecting one equation from those which have not been eliminated. The equation which best catches the turning points and whose recent accuracy is best, generally provides the best basis for forecasting. This criterion will usually be sufficient to select the equation which will be the winner.

2. Ex Ante Tests

   Estimating an equation over part of the historic data and using it to forecast the remainder of the historic interval provides a means of checking ***forecasting*** accuracy. Since the forecasts are made over an interval beyond that used to estimate the equation, the forecasts are called ex ante. With this test, the equation whose estimated historic values are closest to the *ACTUAL* historic results is superior. This test is generally required only when an examination of turning points and recent accuracy fails to identify a clearly superior choice.

3. Best *DURBIN-WATSON STATISTIC*

   Given equations which survive all previous tests of comparability, the equation with the *DURBIN-WATSON STATISTIC* closest to 2.0 can be a basis for selection. Usually the previous criteria will have been sufficient to make a choice among equations. This test is last because it is better to have an equation with a smaller error (i.e. higher *R-BAR SQUARED* or lower *STANDARD ERROR OF THE REGRESSION*) which persists (i.e. poorer *DURBIN-WATSON*), than a larger error which is random. If all else fails, autocorrelated errors can be phased out mechanically (see the end of the discussion of the Pitfall of *AUTOCORRELATION*).

---

[1] A book that is often used to explicate these concepts in a more rigorous fashion is J. Johnston's *Econometric Methods* published in a 2nd Edition in 1972 by McGraw-Hill Book Company. Klein, L. R., *An Introduction to Econometrics*, Prentice Hall, 1962 is a good reference for elementary study. Wonnacott, R. J. and Wonnacott, R. H., *Econometrics*, Wiley, 1970 is often recommended for intermediate study. Finally, Theil, H., *Principles of Econometrics*, Wiley, 1971 is a standard advanced text.

F STATISTIC

The F statistic tests the null hypothesis that the coefficients of the independent variables are equal to zero. If the hypothesis is supported this would indicate that there is no important relationship between the independent variables and the dependent variables. If the F statistic does not support the hypothesis, then this would indicate that there is an important and predictive relationship between one or more of the independent variables and the dependent variable. The t-statistic can then be used to select out which individual independent variables are most highly related to the dependent variable.

A low F value is considered to be supportive of the null hypothesis and a high F value is considered to be counter-indicative of the null hypothesis.

The F value is supplied with its degrees of freedom and an F table in a statistics book should be consulted to find the appropriate F level by which to judge whether or not your F value is high or low (i.e., significant or non-significant).

The F statistic is only supplied by Tiny Troll when more than two variables are used in the REGRESSION command.

VARIANCE-COVARIANCE MATRIX

The variance-covariance matrix describes the significance and interelationships of the coefficients. Multiple linear regression assumes that the coefficients are distributed joint normally with the relevant parameters equal to the variance-covariance matrix. An example of interpreting the matrix is given below:

CFT	0	1	2
0	.45	-.21	.49
1	-.21	.62	.01
2	.49	.01	.33

Note that this matrix is symmetric and positive-definite. The terms along the diagonal are the variance of our estimates. Thus, for coefficient 1, the standard error of our estimate is the square root of .62. The covariance of 1 with 2 is .01 which means that there is essentially no relationship between the two coefficients. If there is a strong relationship between all the coefficients, this is sometimes referred to as multi-collinearity which means that our estimates of the coefficients are rather diffuse.

APPENDIX A

ARITHMETIC, TRIGONOMETRIC, AND LOGICAL FUNCTIONS USEABLE WITH THE DO COMMAND

ARITHMETIC FUNCTIONS

+ : Addition
- : Subtraction
/ : Division
* : Multiplication
^ : Exponentiation
LOG(n) : Natural logarithm of n
EXP(n) : e raised to the nth power
SQR(n) : Square root of n
INT(n) : Integer value of n
ABS(n) : Absolute value of n
SGN(n) : Sign of n
RND(n) : Random value between 0 and n

TRIGONOMETRIC FUNCTIONS

ATN(n) : Arctangent of n
COS(n) : Cosine of n
SIN(n) : Sine of n
TAN(n) : Tangent of n

LOGICAL FUNCTIONS

< : Less than
> : More than

= : Equal to
<= : Less than or equal to
>= : More than or equal to
<> : Not equal to
AND : Logical "and"
OR : Logical "or"
NOT : Logical "not"

Any arithmetic, trigonometric, or logical function in Applesoft BASIC, is legal in the right hand side of the DO command of Tiny Troll. The only difference is that, instead of using numberic data as part of the argument, you should use the name of the file you are transforming and make sure that it is surrounded by empty spaces.

APPENDIX B

ERROR MESSAGES

Note: Tiny Troll recovers from all error messages but aborts the command on which the error occurred.

ERROR MESSAGE	DESCRIPTION
ACTIVE FILE	Indicates that attempt was made to name a new file with the name of a file already active.
APPLESOFT OR DOS ERROR	This message will not normally occur. If it does, simply ignore it and re-enter your command.
BAD DATE SPEC	You used a date in DEDIT or RANGE which didn't exist in file.
BAD NUMERIC VALUE	You used a number in a command which didn't exist in file.
BAD PERIODICITY	Indicates you either attempted to define a file with a periodicity 0 or attempted to use files with different periodicities in REGRESSION, PLOT, or LIST.
BAD RANGE	You attempted to PLOT or REGRESSION files with different RANGEs.
BAD YEAR	You attempted to use a year which wasn't in the file.

ERROR MESSAGE	DESCRIPTION
DIVISION BY ZERO ERROR	At some point in the calculation of a DO, STATISTICS, or REGRESSION command, there occurred a division by zero, terminating execution.
NOT ACTIVE FILE	Attempt was made to use or alter file which isn't active in Tiny Troll.
SYNTAX ERROR	Most common Tiny Troll error message. Usually indicates that one or more of the Tiny Troll syntax rules were not followed.

RECOVERY FROM ACCIDENTAL RESET

If you accidentally hit the reset button, you do not necessarily lose your active files. Just type the following:

```
* 3D0G
] GOTO 170
```

and the command prompt will be returned to you with your files intact. You will, however, have lost the ability to SAVE your diles onto diskette or LOAD files from diskette.

GETTING OUT OF TINY TROLL

If, for any reason, you want to stop the running of Tiny Troll (e.g., the REGRESSION command is taking too long to be completed), just hit the CTRL key and the C key at the same and hit return and you will be exited from Tiny Troll. You can get back into Tiny Troll by typing GOTO 170, as you did in accidental reset. You will, however, also lose the ability to SAVE, LOAD, and CATALOG.

Note: Tiny Troll has been carefully pre-tested to eliminate "bugs" but, as in the case with all new software packages, there are likely to be a few undetected problems which it may be your misfortune to discover. If this should happen to you, please don't hesitate to contact us so that we can correct the problem. Thank you.

APPENDIX C

SUMMARY OF TINY TROLL COMMAND SYNTAX STRUCTURES

Tiny Troll is a language and, thus, has a number of syntactic rules which are general to the system. The two most important rules are that each word in a command must be separated by an empty space, and that each complete command must be terminated by a space followed by a colon.

Each of the individual commands in Tiny Troll has its own syntactic formula which will be listed below. The representations of these various syntactic formulas do follow a general format:

```
KEYWORD argument [argument ...] :
```

The KEYWORD is a word from the command menu or a sub-command of the DEDIT command. Whenever a word in all capitals is encountered in a syntactic formula, it is important that the word be typed in exactly as is in the command (it can, however, be shortened to a two letter abbreviation). A lower case "argument" is to be replaced in the command by a date, data value, or filename. If an argument is enclosed in brackets, this indicates that the argument is optional. If the argument in brackets is followed by an elipses, the argument can be entered repeatedly into Tiny Troll in the same command.

DEDIT

SYNTAX: DEDIT [NEW] filename [periodicity start-year start-period] :
EXAMPLE: DEDIT NEW HELLO 2 75 1 :

DEDIT SUB-COMMANDS:

SYNTAX: ADD date value [value ...] :
EXAMPLE: ADD BOTTOM 18.2 19.5 17.2 :
SYNTAX: ERASE date : (to ERASE one value)
EXAMPLE: ERASE TOP :
SYNTAX: ERASE date date : (to ERASE all the values between the two dates)
EXAMPLE: ERASE 65:4 68:2 :

DO

SYNTAX: DO filename = expression :
EXAMPLE : DO JUNK = PIE /SIN(ICECREAM) :

HELP

SYNTAX: HELP :

LIST

SYNTAX: LIST filename [filename ...] :
EXAMPLE: LIST JUNK PIE ICECREAM :

NAMES

SYNTAX: NAMES :

NAME

SYNTAX: NAME old-filename new-filename:
EXAMPLE: NAME JUNK JEWELS :

PR#

SYNTAX: PR#n : (can also be used in DEDIT mode)

EXAMPLE: PR#1 :

RANGE

SYNTAX: RANGE filename start-year start-period end-year end-period :

EXAMPLE: RANGE TOMMY 68 2 79 7 :

SPECIFICATIONS

SYNTAX: SPECIFICATIONS filename $\substack{\text{YES}\\\text{NO}}$ periodicity start-year start-period : (CHANGE of CH can be used instead of YES.)

EXAMPLE: SPECIFICATIONS PIZZA CH 4 25 1 :

CATALOG

SYNTAX: CATALOG :

LOAD

SYNTAX: LOAD filename :

EXAMPLE: LOAD RESOURCE :

SAVE

SYNTAX: SAVE filename :

EXAMPLE: SAVE CONTACT :

SLOT

SYNTAX: SLOT n : (n must be between 1 and 7)

EXAMPLE: SLOT 7 :

DRIVE

SYNTAX: DRIVE n : (n must be either 1 or 2)

EXAMPLE: DRIVE 2 :

STATISTICS

SYNTAX: STATISTICS filename :

EXAMPLE: STATISTICS TOMATO :

PLOT

SYNTAX: PLOT [ZERO] filename [filename ...] :

EXAMPLE: PLOT HOWDY TV DOODY :

REGRESSION

SYNTAX: REGRESSION filename filename [filename ...] :

EXAMPLE: REGRESSION GNP CON TAXRATE :

NOTE 1:

Any command can be reduced to a two letter abbreviation in the syntax. For example, PLOT HELLO : can be entered as PL HELLO : .

NOTE 2:

An asterisk (*) can be substituted in any command for the name of the file last used in a Tiny Troll command. This simply allows you to avoid retyping the filename of a file you are using continuously in Tiny Troll.

APPENDIX D

CUSTOMIZING

I. MODIFYING TINY TROLL

While we have tried to make Tiny Troll as flexible as possible, we realize that situations may arise in which you would like to modify the program for a specific purpose. Below we have tried to give some suggestions for handling three possible situations. However, we do not encourage you to modify Tiny Troll in any other way. Micro Finance Systems will not support any versions of Tiny Troll which have been modified. The structure of Tiny Troll's code is quite complex, and the amount of free space for extra code is small. Most attempts to alter Tiny Troll will result in making the system dysfunctional.

A. If you have a two disk system, you may wish to run Tiny Troll using both drives. You can use Tiny Troll with a disk controller in any SLOT. When the program comes up initially, it expects that data files will be LOADed from the same SLOT and DRIVE which the program was initially booted from. Using the SLOT and DRIVE commands, you may alter the data DRIVE and SLOT at will. However, you may also wish to permanently change Tiny Troll to initially expect data to come from a different SLOT and

DRIVE than the one you booted from. To do so, boot the system and type END as your first command to return to Applesoft. Now, change line 50040 of the program so the variables SL and DR have the values you want. Then SAVE TINY TROLL. For example, suppose you boot from SLOT 6, DRIVE 1, but you want the data drive to be initially set to SLOT 6, DRIVE 2. Simply follow the procedure outlined and change the value of DR to 2.

B. Presently TINY TROLL can hold approximately 1000 data values in memory at any one time. By default, the number of variables (files) is set to 9, and the number of cases or values per file is 100. Should you want to change this (to have more shorter files or fewer longer ones), change the value of variable VM at line 50010 to the number of variables you want, and/or the value of CM to the maximum number of cases. Note: the product of VM and CM should be less than 1000. Otherwise you may run out of memory. Also, remember to add five to the number of cases you want before changing CM. The extra five values are for file header information.

C. You may wish to include a special printer driver routine for hard copy output. Use lines 2320-2390 to do this. However, you're strictly on your own here. Tiny Troll's PR# command only issues the PR# command.

II. DATA FILE FORMAT

You may wish to use data files generated by Tiny Troll for another application, or you may wish to input externally generated data in Tiny Troll. This section describes the disk file format of Tiny Troll data files to help you to do either of these.

Essentially, you will need to write a separate program to convert an existing file to Tiny Troll format and SAVE it to disk. Then you can use Tiny Troll in the normal fashion.

Tiny Troll expects data to be DOS 3.2 text files with the following format:

Record 0=number of data values in the file (this would be 4 less than total number of records in the file -- the first 4 records form the file header.)

Record 1=the periodicity
Record 2=the start year
Record 3=the start period
Record 4=the data values

The data file format uses fixed-length records of length seventeen.